RESEARCH PAPER | Computer Science | Volume : 3 | Issue : 5 | May 2013 | ISSN - 2249-555X

Query Planning of Continuous Aggregation Queries Over Network

KEYWORDS: Algorithm, coherency, continuous queries, data dissemination, distributed query processing, performance

Prof. Y. B. Gurav
Kothrud, Pune

Mrs. Manjiri Deshmukh
Kothrud, Pune

ABSTRACT *To monitor the time varying data we required some queries and those queries are continuous queries which provides result 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. This paper presents 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 web-page are served by one or more nodes of a content distribution network (CDN) 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. 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, Build a query cost model which can be used to estimate the number of messages required to satisfy the client specified incoherency bound.*

I.INTRODUCTION

A. Overview

The web is becoming a universal medium for information publication and use. Such information is often dynamic, allowing users to employ it for online decision making. Data delivered over the internet is distributed and is increasingly being used for personalized experiences. Typically user desires to obtain the value of some function over the distributed data items. Given the increasing number of applications that make the use of highly dynamic data, there is significant interest in systems that can efficiently deliver relevant updates automatically. For this scenario, answering user queries requires acquiring dynamic data values from distributed sources and aggregating those data values to satisfy user's requirement. Because these queries involve aggregation of dynamic data items, query results must be refreshed continuously so that users get the updated notifications of the data. These are long running queries which allow the data to be delivered to the user over fast changing data from distributed sources executed over data aggregators leading to significant improvement in use of network and resources.

Data Aggregators can be called as organizers involved in compiling information from detailed database on individuals and selling information to others. For online purpose where dynamic data is of prime importance, data aggregators can gather the information from designated websites and providing the data to the user. The process of extracting raw statistical information from the database or data repository, putting it all together to produce statistical output that can be used by the user and has relevance to statistical query it seeks to satisfy. Absolute difference in the value of data item at the data source and the value known to the client.

Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. This paper," Query planning for Continuous Queries over a Network of data Aggregators", presents a low-cost, scalable technique to answer continuous aggregation queries using a content distribution network of dynamic data items. It saves the time and the user spending low cost. A continuous query cost model which can be used to estimate the number of messages required to satisfy the client specified incoherency bound and optimal query plan for continuous aggregation queries.

II. LITERATURE SURVEY

Web sites become popular, they're increasingly vulnerable to the flash crowd problem, in which request load overwhelms some aspect of the site's infrastructure, such as the front end Web server, network equipment, or bandwidth, or (in more advanced sites) the back-end transaction-processing infrastructure. Our approach is based on the observation that serving Web content from a single location can present serious problems for site scalability, reliability, and performance. We thus devised a system to serve requests from a variable number of surrogate origin servers at the network edge. Running applications on a globally distributed network of computers provides many of the same advantages as simple content delivery: capacity on demand, cost-effective use of shared resources, ability to respond to users without communicating over long distances, and so on[3].

As Internet traffic continues to grow and web sites become increasingly complex, performance and scalability are major issues for web sites. Web sites are increasingly relying on dynamic content generation applications to provide website visitors with dynamic, interactive, and personalized experiences. However, dynamic content generation comes at a cost - each request requires computation as well as communication across multiple components. To address these issues, several back end caching approaches have been proposed, including query result caching and fragment level caching. The paper, presents an approach and an implementation of a dynamic proxy caching technique which combines the benefits of both proxy-based and back end caching approaches. The results of this implementation indicate that our technique is capable of providing order-of magnitude reductions in bandwidth and response times in real-world[2].

III. NETWORK OF DATA AGGREGATOR (DA)

A. Data Aggregator

The required data updates for query evaluation at a DA can be obtained either by sources pushing them continuously or the DA pulling them whenever required. In both these cases, it is desired that client's coherency and fidelity requirements

Volume : 3 | Issue : 5 | May 2013 | ISSN - 2249-555X

are met with the minimum number of data value refresh messages. Thus efficiency of query evaluation at the DA is quantified in terms of the number of refreshes done for various data items involved in the queries executing at the DA. Reducing the number of refreshes reduces computational overheads at the DA, load on the data servers and network bandwidth. Since data sources have exact data values, we can expect that a push based approach can deliver the required bounded incoherency with the given fidelity using a smaller number of refresh messages.

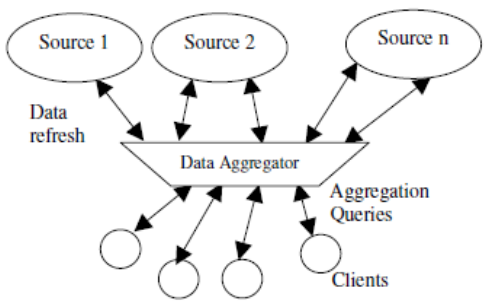

Fig 1 Query execution at data Aggregators

B. Data incoherency

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 paper [5]. 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. Data accuracy can be defined in terms of incoherency of a data item, defined as absolute difference between the value id data at the data source and value known to the client of a data. Let v(t) denote the value of ith data item at the data source at a time t; and let value the data item known to the client be u(t). Then the data incoherency at the client is given by $|v(t)-u(t)|$.

1) Aggregate Queries and Their Execution : Consider a client query Q=50d1+200d2+150d3, where d1, d2, d3 are different data items with a required incoherency bound of $80. We want to execute this query over the data aggregators to minimize the number of refreshes. 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 d1, d2, and d3 separately. The query incoherency bound can be divided among data items in various ways ensuring that query incoherency is below the incoherency bound. 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 (dq= 50d1 + 200d2 + 150d3) and disseminate the result to the client so that the query incoherency bound is not violated. A third 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 multiple sub queries since different data aggregators disseminate different subsets of data items, no data aggregator may have all the data items required to execute the client query 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.

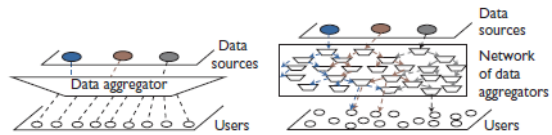

Fig 2 (A)Single data Aggregator (B)Network of Data Aggregators

3)Weighted Additive Aggregation Query : Value of a continuous weighted additive aggregation query, at time t, can be calculated as: $Vq(t)=\sum (Vqi(t)*Wqi)$. Where Vq is the value of a client query q involving nq data items with the weight of the ith data item being $wqi, 1 \le i \le nq$. Suppose the result for the query given by (1) needs to be continuously provided to a user at the query incoherency bound Cq. Then, the dissemination network has to ensure that- $|\sum (Vqi(t)-Uqi(t))*Wqi| \le Cq$ 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.

The problem of choosing sub queries while minimizing query execution cost is an NP-hard problem. Efficient algorithms are given to choose the set of sub queries and their corresponding incoherency bounds for a given client query. For solving the above problem of optimally dividing the client query into sub queries, we first need a method to estimate the query execution cost for various alternative options. As we divide the client query into sub queries such that each sub query gets executed at different aggregator nodes, the query execution cost (i.e., number of refreshes) is the sum of the execution costs of its constituent sub queries.

Thus query Q can be divided in two alternative ways:

Plan 1. Result of sub query 50d1+150d3 is served by a1, whereas value of d2 is served by a2.

Plan 2. Value of d3 is served by a1, whereas result of Sub query 50d1 + 200d2 is served by a2.

IV DATA DISSEMINATION COST MODEL

Here the model is presented 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.

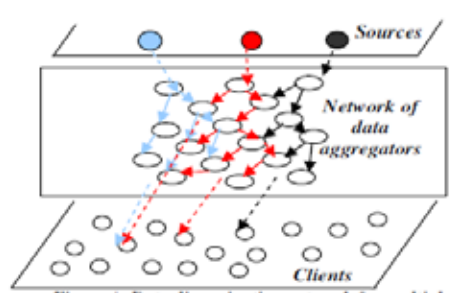

Fig.3:Data Dissemination of Multiple Data item

A. Incoherency Bound Model

Incoherency bound model is used for estimating dependency of data dissemination cost over the desired incoherency bound. As per this model, the number of data refreshes is in-

Volume : 3 | Issue : 5 | May 2013 | ISSN - 2249-555X

versely proportional to the square of the incoherency bound (1/C2).. Data dissemination cost $\propto$ 1/C2

B.Data Dynamics Model
There are 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. It is calculated by summing up the products of the deviations of the data item values from their mean that varies between -1 and 1 with higher (absolute) values signifying that data points can be considered linear.

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.

Specifically, the cost of data dissemination for a data item will be proportional to data sumdiff defined as

Rs= $\sum$ |Si-Si-1| (4)

C.Combining Data Dissemination Models
Number of refresh messages is proportional to data sumdiff Rs and inversely proportional to square of the incoherency bound ($C^{2)}$. 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 ($1/C^2$). Thus, for a given data item S, disseminated with an incoherency bound C, the data dissemination cost is proportional to Rs/C^2. In the next section, we use this data dissemination cost model for developing cost model for additive aggregation queries.

D. Cost Model For Additive Aggregation Query
Consider an additive query over two data items P and Q with weights wp and wq, respectively; Number of pushes versus data sumdiff estimate its dissemination cost. If data items are disseminated separately, the query sumdiff will be

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

Instead, if the aggregator uses the information that client is interested in a query over P and Q (rather than their individual values), it creates and pushes a composite data item (WPp+WQq) then the query sumdiff will be

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

Where Rquery is clearly less than or equal compared to Rdata. Thus, we need to estimate the sumdiff of an aggregation query given the sumdiff values of individual data items (i.e., Rp and Rq). Only data aggregators are in a position to calculate Rquery as different data items may be disseminated from different sources.

V. 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 ak €A(1 ≤k ≤N), set Dk is subset of D of data items disseminated by the data aggregator ak, and incoherency bound tkj which the aggregator ak can ensure for each data item dkj €Dk.

2. Client query q and its incoherency bound Cq. An additive aggregation query q can be represented as∑ wqi dqi, where wqi is the weight of the data item dqi for 1 ≤i≤ nq.

Outputs:
1. qk for 1≤ k≤ N, i.e., sub query for each data aggregator ak.

2. Cqk for 1≤ k ≤N, i.e., incoherency bounds for all the sub queries.

Thus, to get a query plan we need to perform following tasks:

1. Determining sub queries: For the client query q get sub queries qks for each data aggregator.
2. Dividing incoherency bound: Divide the query incoherency bound Cq among sub queries to get Cqk s.

For optimal query planning, above tasks are to be performed with the following objective and constraints: Optimization objective. Number of refresh messages is minimized. , as it is proved that, for a sub query qk, the estimated number of refresh messages is given by

$-Rqk/C^2qk$, where Rqk is the sumdiff of the sub query qk; Cqk is the incoherency bound assigned to it and k, the proportionality factor, is the same for all sub queries of a given query q. Thus, total number of refresh messages is estimated as

$$Z_q = \kappa \sum_{k=1}^{N} \frac{R_{qk}}{C_{qk}^2}.$$

Hence, Zq needs to be minimized for minimizing the number of refreshes.

A.Greedy Heuristics for Deriving the Data-items
Here is given the outline of greedy algorithm for deriving data-items. First, we get a set of maximal data-items (Mq) corresponding to all the data aggregators in the network.

The maximal subquery for a data aggregator is defined as the largest part of the query which can be disseminated by the DA (i.e., the maximal subquery has all the query data items which the DA can disseminate). For example, consider a client query 50d1 + 200d2 +150d3. For the data aggregators a1 and a2 given in Example 1, the maximal subquery for a1 will be m1=50d1 + 150d3, whereas for a2 it will be m2 = 50d1 + 200d2. For the given client query (q) and relation consisting of data aggregators, data items, and data incoherency bounds (f(A; D;C)) maximal data-items 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. This operation can be performed in O(|q|:max|Dk|) where|q|is number of data items in the query, max|Dk| is the maximum number of data items disseminated by any DA. For each subquery m €Mq, its sumdiff Rm can be calculated.

1) 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_(S, C)) maximal sub-queries can be obtained or each data aggregator.

Let A be the set of such maximal data-items. In this set, each query a €A can be disseminated by a designated data aggregator at the assigned incoherency bound. For each subquery a €A, its Sumdiff Ra is calculated using Equation.

Using the set A and sub-query sumdiffs, we use the algorithm to get the set of sub-queries minimizing the query cost.

Each sub-query a €A is represented by the set of data items covered by it. As we need to minimize the query cost, a data-item with minimum cost per data item is chosen in each iteration of the algorithm i.e., criteria minimize (Ra/Ca2|a|).

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

Volume : 3 | Issue : 5 | May 2013 | ISSN - 2249-555X

THE GREEDY ALGORITHM FOR QUERY PLAN SELECTION
Result □ Ø
While A ≠ Ø
Choose a data-item a □ A with criteria ψ
Result □Result υ a
A □ A-{a}
For each data element e □ a
For each b □ A
b □ b-{e}
If b = Ø
A □ A-{b}
Else
Calculate sumdiff for modified b
Return Result
Fig.4.Greedy Algorithm.

VII. QUERY PLAN

A. Overheads of Query Plan

Here the time overheads for various query planning operations is reported. We measured these costs by varying the number of data items being disseminated by the network, between 40 and 200.

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.

Higher cost of query planning, for the sumdiff-based algorithms, is justified by the savings we achieve in terms of number of messages for the whole duration of the continuous query. The query planning cost of random and minCost is higher as they require more iterations of the algorithm (i.e., more data-items) compared to the maxGain algorithm.

A. Optimal Query Planning Problem Is NP-Hard

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. In the set covering optimization problem the task is to find a set covering which uses the fewest sets. We can map the set cover problem to our query planning problem.

The MAX query, corresponding to the set cover problem, will be max of all the items in the universe U at an incoherency bound 1. For each set s 2 S we assume the existence of a DA disseminating all the elements of s at an incoherency bound of 1.

Further, let all data items have sumdiff value of 1. we can see that cost of any subquery will be 1. Thus, cost of the client query, which is sum of cost of its data-items, will be same as the number of subsets required to get the set cover. It is easy to see that if we can solve the query planning problem optimally we can also solve the set cover problem optimally. Thus, now we give greedy heuristics for the data-items selection problem.

To execute the query using a network of data aggregators, subqueries are assigned to different DAs. Each sub query is a query over a subset of query data items. For optimal planning ther is a 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 the need to minimize sum of subquery sumdiff values.

Thus use of greedy algorithm is given for solving the query planning problem with different set of subquery selection criteria ð. In the min-cost heuristic we select the subquery having minimum subquery sumdiff per data item. For the MAX query, subquery sumdiff is nothing but the sumdiff of the most dynamic data item in the subquery. Thus, for the max-gain heuristic, the gain of each subquery is calculated as

$$G = \sum_{d_{qi}} R_{diq} - \max(R_{diq}),$$

Thus for the additive queries with one difference the max-gain algorithm works better than the min-cost algorithm but, unlike in additive queries, in case of MAX queries performance of min-cost algorithm is closer to that of the random algorithm compared to the max-gain algorithm and can be ensured that the most dynamic data item is a part of lower cost subquery, leading to a better query plan.

VIII. CONCLUSION

This seminar presents an approach to minimize the number of refreshes required to execute an incoherency bounded continuous query. It is assumed 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. An important measure for data dynamics in the form of sumdiff is developed which, as we discussed in previously, is a more appropriate measure compared to the widely used standard deviation based measures. For optimal query execution, the query is divided into data-items and evaluate each subquery at a judiciously chosen data aggregator.

REFERENCE
[1] Rajeev Gupta and Krithi Ramamritham, "Query planning For Continuous Aggregation Query Over A Network Of Data Aggregators," IEEE Knowledge and Data Engineering, Vol. no.24, no. 6, June 2012. | [2] 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. | [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] 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 | [5] 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. | [6] 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. | [7] Rajeev Gupta and Krithi Ramamritham," Scalable Execution of Continuous Aggregation Queries over Web Data". Content of IEEE Internet Computing, January/February 2012 | [8] 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. | [9] " Query Cost Model Validation for Sensor Data,"www.cse.iitb.ac.in/~grajeev/sumdiff/RaviVijay_BTP06.pdf, 2011. | | [10] C. Olston, J. Jiang, and J. Widom, "Adaptive Filter for Continuous Queries over Distributed Data Streams," Proc. ACM SIGMOD Int Conf. Management of Data, 2003.