Execution Planning for Continuous Queries over Dissemination Network of Data Aggregator

M .Naresh Kumar

Sri Sai Aditya Institute of Science & Technology,

Surampalem, India

R.Sailaja

Sri Sai Aditya Institute of Science & Technology,

Surampalem, India

ABSTRACT

Normally continuous queries are those used, to observe the dynamically changing data and to give results helpful for Online decision making. Generally a user wants to achieve the value of few aggregation functions over shared data items, for example, to regulate when the value of a stock portfolio exceeds a threshold. We focus on approaches and techniques to assign such dynamically changing data to a large number of users with high accuracy, efficiency, and scalability. In these queries a client maintains a consistency requirement as part of the query. We come up with a minimal –cost based approach to answer continuous aggregation queries using a network of aggregators of dynamic data items. In that network of data aggregators, each data aggregator handles a set of data items at definite coherencies. Our technique is splitting a client query into sub-queries and executing sub-queries on properly chosen data aggregators with their respective sub-query incoherency bounds. We give a mechanism for collecting the optimal set of sub-queries with their inconsistency bounds which satisfies client query's coherency requirement with minimum number of update messages issued from aggregators to the client. We layout a cost based model which can be used to evaluating the number of messages prescribed to satisfy the client specified incoherency bound .

Key Words— Coherency, Continuous queries, Cost, Distributed query processing, Data dissemination, Performance.

1. INTRODUCTION

Web applications demand is increasing rapidly for Data Engineering in presently. Data (i.e., Information) is fundamental thing in dynamically changing Web applications. While Keep tracking of data, we figure out the problem of intensifying and executing multiple continuous queries, where each query is a combination of filters and each filter may appear in various queries. When filters are expensive, significant performance gains are achieved by sharing filter evaluations across queries.

Data Aggregators can be consider as alternate servers in the computation of results from detailed database .For performing online analysis dynamic data is prominent thing. Data aggregator collects the information from designated databases and giving the data to the user.

The comprehensive use of sensor networks in scientific and engineering applications leads to increased demand on the efficient computation of the collected sensor data. Recent research in sensor and stream data systems adopts the notion of sliding windows to process continuous queries over infinite sensor readings. Ordered processing of input data is essential during query execution for many application scenarios. In this paper we discuss three approaches for ordered execution of continuous sliding window queries over sensor data. *The first approach implements* ordered processing at the input side of the query execution plan. In the *second approach* we utilize the advantage of out-of-order execution to optimize query operators and enforce an ordered release of the output results. *The third approach* is adaptive and switches between the first and second approaches to achieve the best overall performance with current input arrival rates and level of multiprogramming. We study the performance of the proposed approaches analytically and experimentally and under a variety of conditions such as the asynchronous arrival of input data, and various levels of multiprogramming. Our performance study is based on an extensive set of experiments using a realization of the proposed approaches in a prototype stream query processing system.

Content dissemination networks (CDNs) approach figure out the problem for static content applying caches at the networks of edge nodes. CDNs continue to appear to handle more and more dynamic applications. A dynamically generated web page is normally assembled applying a number of static or dynamically generated fragments. The static fragments are served from the regional caches whereas dynamic fragments are generated either by accepting the cached information or by retrieving the data items from the origin data sources. One essential question for satisfying client requests through a network of nodes is a way to choose the impressive node(s) to satisfy the client request. In dynamic CDNs, while choosing the node(s) as per the client demand, the central application (top-level CDN node) to make sure that page/data served meets client's coherency essentials too. Techniques to efficiently serve rapid changing data items with guaranteed incoherency bounds have been proposed in the literature. That dynamic data dissemination networks will help to distributed information like stock quotes, temperature data from sensors. In this paper we propose a method to efficiently answer aggregation queries involving such data items.

2. PROPSED SYSTEM

In this paper our focus is, to present a low-cost, scalable technique to answer continuous aggregation queries using a network of aggregators of dynamic data items. In that network of data aggregators, each data aggregator handles a set of data items at specific coherencies. Our approach involves disintegrate a client query into multiple sub-queries and executing sub-queries on relevantly chosen data aggregators with their individual sub query incoherency bounds.

We address a method for getting the optimal set of sub-queries with their incoherency bounds, which satisfies client query's coherency requirement with less number of refresh messages sent from aggregators to the client. . We design a cost based model which can be used to evaluating the number of messages prescribed to satisfy the client specified incoherency bound .

Our objective 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 attained 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 continuously.
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.

3. OPTIMAL QUERY PLAN

We determine that the issue of selecting sub queries while decreasing query execution cost is an NP hard problem. Optimal Query Plan is NP-hard for proving that the problem is NP-hard, For a given client query and DA, let us first define maximal sub-query as the largest part of the query which can be disseminated by the DA (i.e., the maximal sub-query has all the query data items which the DA can disseminate at the required incoherency bound). For example, for a client query 30*S1* +45*S2*+75*S3* with incoherency bound 150; let the pre-decided incoherency bound for each data item be 1. For the data aggregators *D1* and *D2* given in *Example 1*, the maximal sub-query for *D1* will be *q1*=30*S1* +75*S3*, whereas for *D2* it will be *q2*=30 *S1* + 45*S2*.

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: Split the aggregation query into sub queries; and

Task 2: Allocate the query incoherency bound among them. While satisfying the following conditions.

Condition 1. Query incoherency bound is satisfied.

Condition 2. The chosen DA should be able to provide all the data items appearing in the sub query assigned to it.

Condition 3. Data incoherency bounds at the chosen DA should be such that the sub-query incoherency bound can be satisfied at the chosen DA.

Objective: Number of refreshes should be minimized.

Data Incoherency

The objective of incoherency bound model is for estimating dependency of data dissemination cost over the desired incoherency bound.

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.

Network of data aggregators

Data aggregators are one kind of secondary server it serves as data sources (data items). 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.

Security Model

This model is used to help the user to provide the security of access. Because once the user to logout or leave our account automatically user password is changed and server to send the password in our mail ID. Whenever the user to logout the account automatically the security key is changed based on the random function.

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, sum diff, 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.

Client/Server Module

This model is used to help the client and server interaction to the database. It is used to dynamically create the table based on the server entering value. These values are assigning to the chat x and y position and display the client. These values are changed in dynamically based on the server entering values.

4. ALGORITHM USED

The greedy method is the most advisable method. It is prominent for achieving the optimized solutions.

Figure 1 provides the outline of greedy algorithm

```
Solve the  n sizing  sub problems
 if the solution found is better than current lower bound
      then update the current lower bound  and let cont = 0 ;
      else    let   cont = cont +1;
Calculate  the  Subgradient
if the  solution obtained  is suitable
         then update the upper bound
if the  solution  obtained is optimum  then  STOP
else
if  the current solution is suitable
     then apply the improvement heuristic
```

for extracting sub queries.

Fig 1: Greedy Algorithm for Query Plan Selection

5.OPTIMAZATION CHALLENGES

We address the problem of finding the optimal shared execution strategy for any given collection of queries that share expensive filters. Finding the optimal shared execution strategy poses the following major challenges:

1. **Filter placement**. The decision whether a filter should be evaluated earlier or later in a shared strategy should be made by taking all of the following factors into account:

• **Cost:** Filters with low cost should preferably be evaluated early, since they might resolve queries at lower cost.

• **Selectivity:** The average fraction of incoming data items that satisfy a filter is referred to as the *selectivity* of that filter. Filters with lower selectivity should preferably be evaluated early, since they are more likely to resolve queries by evaluating to false.

• **Participation:** The number of queries that contain a given filter is referred to as the *participation* of that filter. Filters with higher participation should preferably be evaluated early, since they can decide the results of a larger number of queries. For a given filter, these three factors may give contradictory suggestions for its placement, so we must devise a placement method that takes all the factors into account.

2. **Execution Overhead**. Executing a shared strategy incurs some amount of overhead, e.g., keeping track of which filters have been evaluated, and which queries have been resolved. This overhead is in addition to the cost of the filters evaluated by the strategy. Thus the overall choice of the optimal strategy must take into account the expected total cost of filters evaluated by a strategy as well as its execution overhead.

5.1 Delay Factors of query

The following factors are used to determine the preference factor of a node:

1. *Data Availability Factor:* The number of data items at a parent can serve Q with its current data and coherency requirement.

2. *Computational delay Factor:* The larger the computational delay incurred at a parent P to disseminate a data change to its dependents, the less preferred it is. We approximate this delay by the number of dependents P has: On average, the more dependents P has, the greater will be the computational delays encountered by Q to get a data update from P

3. *Communication delay Factor:* Parents which have a large communication delay with Q are less preferred.

5.2 Solve Query and Sub Queries

1 We should be selecting the plan with lesser number of sub-queries. But that is not guaranteed to be the plan with the least number of messages. Further, we should select the sub-queries such that updates to various data items appearing in a sub-query have more chances of canceling each

other as that will reduce the need for refresh to the client (Equation 2). In the above example, if updates to *S1* and *S3* are such that when *S1* increases, *S3* decreases, and vice-versa, then selecting *plan1* may be beneficial. We give an algorithm to select the query plan based on these observations. While solving the above problem of selecting the optimal plan we ensure that each data item for a client query is disseminated by one and only one data aggregator. Although a query can be divided in such a way that a single data item is served by multiple DAs (e.g., 30 *S1* + 45 *S2* + 75 *S3* is divided into two sub-queries 30 *S1* + 75 *S3* and 30 *S1* + 45 *S2*); 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 multiple sources).

3. 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 30*S1* + 75*S3* should be greater than 55 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.

6. RELATED WORK

Distinct mechanisms for efficiently maintain incoherency bounded aggregation queries over continuously changing data items are proposed in the literature [6, 8, 11]. In [9] authors predict that each client's data specifications are satisfied by single data aggregator. In such case, data aggregators may need to disseminate a extensive data items which will lead to huge number of refresh messages, so increase in delay. Hence every client getting complete their data item from a single data aggregator is optimal in view of number of messages. Hence by our work, one can frame anticipated number of messages for the client query. So, by this work can complete the optimizing Fidelity of data items. In [6] authors addressed push based fetch applying data filters at the sources. Based on that, for an aggregation query, the number of refresh messages will be reduced by achieving incoherency bound allocation to respective data items such that the number of messages from different data sources is similar. Alternately we execute more dynamic data items as part of bigger sub quires while optimally assigning incoherency bounds. In [8] authors proposed pull based data dissemination methods, where clients or data aggregators

Pull data items such that query specifications are satisfied. In [17] author presented cost-based methods to create network aggregation tree.

7. CONCLUSION

In this paper, we presented query optimization using cost based technique, to reduce the number of refresh messages required to execute an incoherency bounded continuous query. For optimal execution we divide the query into no of sub queries and estimate each sub-query at a chosen aggregator. 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. Our future work is changing a query plan as data dynamics changes and developing the cost model for more complex queries.

REFERENCES

[1] Rajeev Gupta, Kirthi Ramamrithm, *"Query Planning for Continuous Aggregation Queries Over a Network of data Aggregators "*, IEEE 2012 Transactions on Knowledge and Data Engineering, Vol .24, Issue: 6

[2] A. Davis, J.Praikh and W.Weihl, *"Edge Computing Extending Enterprise Applications to The Edge Of the Internet"*, WWW 2004.

[3] D.VanderMeer, A.Datta, K.Dutta, H.Thomas and Ramamritham, *Proxy –Based Acceleration of Dynamically Generated Content on the World Wide Web"*, ACM Transactions on Database Systems (TODS) Vol.29, June 2004.

[4] S.Rangarajan, S.Mukerjee and P.Rodriguez, *"User Specific Request Redirection in a Content Delivery network "*, 8 Intl .Workshop on Web Content Caching and Distribution (IWCW), 2003.

[5] S.Shah, K.Ramamritham, and P.Shenoy, *"Maintaining Coherency of Dynamic Data in Cooperating Repositories "*, VLDB 2002.

[6] C.Olston, J.Jiang and J.Widom ,*"Adaptive Filter for Continuous Queries Over Distributed Data Streams"*.SIGMOD2003

[7] S Shah, K .Ramamritham, and C.Ravishankar *"Client Assignment in Content Dissemination Networks for Dynamic Data "* , VLDB 2005.

[8] R Gupta , A Puri , and K.Ramamritham , *" Executing Incoherency Bounded Continuous Queries at Web Data Aggregators"*, WWW 2005.

[9] Y.Zhou, B. Chin Ooi and Kian-Lean Tan, *"Disseminating Streaming Data in a Dynamic Environment: An Adaptive and Cost Based Approach "*, The VLDB Journal, Issue 17, Pg.1465- 1483, 2008

[10] S.Madden, M.J.Franklin, J.Hellerstein and W.Hong, *"TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks "*, Proc. Of 5th Symposium on Operating Systems Design and Implementation,2002.

[11] S. Agrawal, K. Ramamritham and S. Shah, *"Construction of a Temporal Coherency Preserving Dynamic Data Dissemination Network "*, RTSS 2004.

[12] A.Iyenger and J.Challenger , *"Improving Web server Performance by Caching Dynamic Data"* . Proceedings of the USENIX Symposium on Internet Technologies and System (USEITS) 1997.

[13] R. Srinivasan, C .Liang, K. Ramamritham, *"Maintaining temporal coherency of Virtual Datawarehouses.* ", Proceedings of the IEEE –Real time Systems Symposium P.60, December 02-04,1998.

[14] A Deshpande, C. Guestrin, S .R. Madden, J. M. Hellerstein; and W. Hong , *"Model-Driven Data Acquisition in Sensor Networks "*, VLDB 2004.

[15] R.Gupta and K.Ramamritham, "Optimized Query Planning of Continuous Aggregation Queries in Dynamic Data Dissemination Networks ", WWW 2007.

[16] N .Jain, D. Kit , P.Mahajan ,P.Yalagandula , M .Dahlin and Y.Zhang ,*" STAR : Self – Tuning Aggregation for Scalable Monitoring "*, VLDB 2007.

[17] P. Edara, A.Limaye and K.Ramamritham, *"Asynchronous In-network Prediction 4 , :Efficient Aggregation in Sensor Networks "*, ACM Transactions on Sensor Networks, Volume Number 4 , August 2008.

.