International Journal of Modern Trends in Engineering and Research
www.ijmter.com
e-ISSN No.:2349-9745, Date: 2-4 July, 2015

Dynamic Query Aggregation at Data Aggregators

Komal D Chopra[1], Tejesh K Niwate[2], Mahammad Shabana [3]
[1]*komal.17chopra@gmail.com*
[2] *Department of Computer Science & Engineering, St. Mary's Engineering College, tejeshkn@gmail.com*
[3] *Department of Computer Science & Engineering, St. Mary's Engineering College,*
mdshabana5824@gmail.com

Abstract— Many applications in today's world uses dynamic data i.e. time-varying data, so there may be some inaccuracy in the result. In some cases inaccuracy is tolerable by client and that is known as incoherency bound. Client in his query specifies the coherency value as his requirement along with the query. The query is given to data source and then executed at specific data aggregators to obtain optimal results. The client query is decomposed into sub-queries and then executed at chosen data aggregator with individual sub-query incoherency bound. To maintain dynamics of data and user's coherency requirement data needs to be updated. This updating of data is done using data dissemination techniques. Two techniques are available pull based and push based data dissemination technique. Here we present push based data dissemination technique for updating data at source and greedy heuristics to divide the query into Sub-queries. We also present techniques to reduce the number of refresh messages.

Keywords- Data aggregator; Decision making; Data Dissemination; Incoherency bound; Network monitoring

I. INTRODUCTION

In today's world user need data which is continuously changing i.e. temperature sensing data, stock value etc. Stock data values are available on different nodes and they need to be aggregated to answer user's requirement. Stock or Temperature sensing values keep on changing continuously and need to be updated. So to keep this values updated we need to refresh the messages and we use data dissemination techniques to update the data at source. Two techniques are present [1]

1. Pull-based data dissemination.
2. 2. Push-based data dissemination.

Here we use push based techniques in which data source sends update message to client on their own. The user must specify his query along with the coherency value.
In Pull-based technique update message is sent to client only when the user requests.

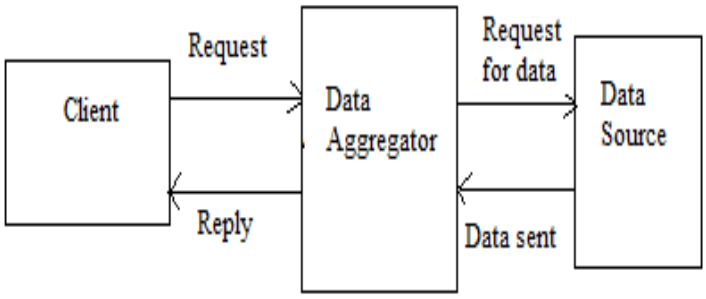

Figure 1. Pull Based Technique

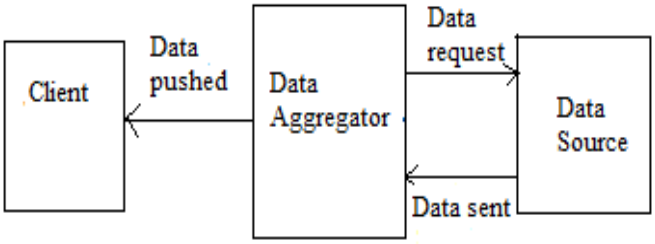

Figure 2. Push based Technique

Data Incoherency: 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 ui(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, |vi(t) - ui(t) |> C. Coherency value is the accuracy requirement user can tolerate. Continuous queries are very essential in data mining as it proved to be very efficient in tracking data. [2]

What is Data Aggregator?
These are the nodes in between data source and client. This node maintains incoherency bound for various data items. Each data aggregator has a set of data item which he can disseminate at an incoherency bound.

Consider a situation to better understand this concept. Scenario: - Consider query Q=55d1+200d2+154d3 Where d1, d2, d3 are data items for stock with incoherency bound of $80. To answer the query given in situation, user can get outcome in three probable ways:-

1. Client can get data items separately. Among data items query incoherency bound is divided.
2. To answer the query a single data aggregator can distribute all data items.
3. A single query can be divided into number of sub-queries and only one data aggregator gives their values.

Numbers of refresh messages are reliant on the division of query incoherency bound. Incoherency of a data item can be defined as difference in value of data item at source and at the node. [3]

II. LITERATURE SURVEY

Shetal Shah Krithi Ramamritham Prashant Shenoy [5] consider techniques for disseminating dynamic data—such as stock prices and real-time weather information—from sources to a set of repositories. Focus on the problem of maintaining coherency of dynamic data items in a network of cooperating repositories. We show that cooperation among repositories—where each repository pushes updates of data items to other repositories—helps reduce system-wide communication and computation overheads for coherency maintenance. They work to improve the fidelity of data stored at repositories.

Rajeev Gupta, Ashish Puri, Krithi Ramamritham in [9] develop and evaluate client-pull-based techniques for refreshing data so that the results of the queries over distributed data can be correctly reported, conforming to the limited incoherency acceptable to the users. We model as well as estimate the dynamics of the data items using a probabilistic approach based on Markov Chains. Authors in this paper make use of existing web infrastructure for answering these queries which leads to minimal architectural requirements and more scalability.

Chris Olston, Jing Jiang, and Jennifer Widom in [11] users register continuous queries with precision requirements at the central stream processor, which installs filters at remote data sources. The filters

adapt to changing conditions to minimize stream rates while guaranteeing that all continuous queries still receive the updates necessary to provide answers of adequate precision at all times.
In [12] author discussed the client assignment problem in a content distribution network for dynamic data.
Many data intensive applications delivered over the Web suffer from performance and scalability issues. Content distribution networks (CDNs) solved the problem for static content using caches at the edge nodes of the networks. CDNs continue to evolve to serve more and more dynamic applications. The static fragments are served from the local caches whereas dynamic fragments are created either by using the cached data or by fetching the data items from the origin data sources One important question for satisfying client requests through a network of nodes is how to select the best node(s) to satisfy the request. For static pages content requested, proximity to the client and load on the nodes are the parameters generally used to select the appropriate node.

III. PROPOSED SYSTEM

In the proposed system, client query along with incoherency bound value is given as input. The client query is sub-divided into number of sub queries and executed at chosen data aggregators. The data aggregators execute the query and give the result to central node. The result from different data aggregators is aggregated at the central node and then given to client as the result. The flow is as shown below.

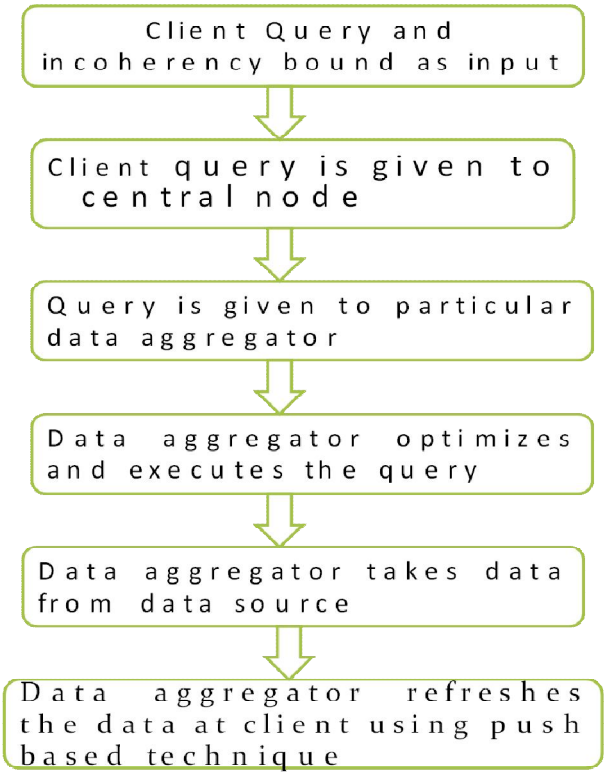

Figure 3. Flow of System

The main goal or aim is to reduce the number of refresh messages. The aim can be achieved by constructing the efficient models. Aim is to satisfy the clients query requirements while minimizing the query execution cost in terms of number of dissemination messages. Towards that end, the following is achieved

1. Developed techniques for estimating the cost of disseminating a data item, at specified incoherency bound.
2. Using the estimated data dissemination cost, develop query cost model for estimating the cost of executing an incoherency bounded continuous query.
3. Used the query cost model for assigning a client query to one or more data aggregators so that the query can be executed with the least number of messages. The work involves dividing the client query into sub-queries and allocating it to different data aggregators for optimal execution.

For query plan that is dividing the query into sub-query Greedy Algorithm is used
Result ← Ø
While A ≠ Ø
Choose a sub-query a ϵ A with criteria ψ
Result ←Result υ a
A ← A – {a}
For each data element e ϵ a
For each b ϵ A
b ← b – {e}
If b = Ø
A ← A – {b}
Else
Calculate sumdiff for modified b
Return Result

IV. MATHEMATICAL MODEL

Input= {Q, I, N}
Output= {R, Sq,Ib}
Where,
Q= Client Query
I= Incoherency bound
N= Network
R= Result
Sq= Sub-queries
Ib= Incoherency Bound of sub-queries
State diagram:
s0 = Initial state where client request query with coherency requirement
s1 =Client query is divided into sub-queries.
s2 = sub-queries are given to specific data aggregators along with subquery incoherency
s3 = Query Planning
s4 = Sub-queries are aggregated and given as a result to client.
s5 = result is received by client

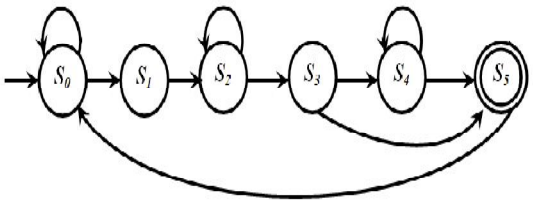

Figure 4. State Transition Diagram

CONCLUSION

The proposed system helps to reduce the number of refresh messages. The system is also capable of balancing the load on different aggregator nodes. A cost model is also presented to show the efficiency of the system.

REFERENCES

[1] Rajiv Gupta and Krithi Ramamritham, "Query Planning for Continuous Aggregation Queries over a Network of Data Aggregators", Fellow IEEE.
[2] Komal Chopra, "Implementation of Query Planning for Distributed Database using Aggregation Queries",IJETAE Volume 3, Issue 3, March 2013.
[3] Suvarna Pawar,Komal Chopra ``Executing Queries over a Network of Data Aggregators in Cloud Network"(IJARET), Volume 1, Issue II ISSN 2320-6802, Mar. 2013.
[4] 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.
[5] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos,"Processing Approximate Aggregate Queries in Wireless Sensor Networks," Information Systems, vol. 31, no. 8, pp. 770-792, 2006.
[6] 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.
[7] 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.
[8] 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.

