Continuous Query Processing Of Dynamic Data Items In Network Aggregation Environment

Prof. P. E. Patel

Dr. K. V. Metre

MET's Institute of Engineering, Nashik

***Abstract:** The web is becoming a universal medium for information publication and usage. Such information is becoming more and more dynamic and usage is varying from simple tracking to online decision making in real time. Examples of such data include information such as network data, currency exchange and stock prices, real-time traffic and weather information and data from sensors in industrial process control applications. Continuous queries are used to get the updates of this time varying data. The continuous queries with an Incoherency requirement are long running queries which continuously give the information. Each data aggregator serves a set of data items at specific coherencies. The client query is decomposed into sub-queries and that sub-queries are executed on chosen data aggregators with their individual sub-query incoherency bounds. A network of data aggregator serves the user with data having options for combinations of aggregators. The goal is to minimize the number of refreshes. The refresh message is sent from the data aggregator to the client. For data update, the data dissemination techniques such as pull based and push-based are used.*

***Keywords:** Continuous queries, Data aggregator, Incoherency, Dynamic data.*

I. INTRODUCTION

The internet is continuously getting loaded with the data. The internet has grown in popularity from being a mere facility to a necessity. It is becoming a universal medium for information publication and usage. The continuous queries are used to monitor the changes to the time varying data and to provide results for online decision making. The client wants to obtain the value of aggregation function over distributed data items. The continuous query is logically run continuously over the space in contrast one-time queries which are run once to over the current data sets.The time-varying data is used for the important decision making. As the data is time varying so the updates should be disseminated to the client continuously. The coherency requirement is dependent on the nature of the data items and user requirement. The examples are stock prices, currency exchange rates, real-time traffic, and weather information, data from sensors, auctions, personal portfolio evaluation, route planning based on traffic information makes extensive use of dynamic data. Data incoherency is the absolute difference in the values of the data items at the source and value known to the client. Let si be the value at the source and ui be the value of the data item at the client. Then the incoherency at the client is given by the | si- ui|. Load balancing can also be done using continuous queries or other network performance adjustments. In financial applications uses continuous queries to monitor trends and detect fleeting opportunities. These two applications are characterized by a need for continuous queries that go well beyond simple element at a time, by fast data streams, and as a need for timely online answers. The example of the stock data is as given below:

Select COMPANY, LISTED_ON_EXCHANGE from STOCK_DATA

This query can be used to collect the stock information from various companies and also the companies listed on the stock exchange. The value of the stock is checked and as soon as the value of the stock is changed, the trigger is generated. The query is divided among the sub-queries and the sub-queries are given to the network of data aggregators.

Aggregators with the help of data source disseminate the result to the client. The client query is divided among the sub-queries and the sub-queries are given to the judiciously chosen data aggregators. The data aggregator which contains maximum part of that sub-query disseminates the result to the client. The query incoherency bound is set by the user among with the query. Incoherency bound is the absolute difference between values of data items at the client and value of data items at the source. Let the value of data item at the client is ui(t) and at data source, it is si(t). Then the data incoherency bound is given as |si(t)- ui(t)|. The data refresh message is sent to the client as soon as, the data incoherency bound exceeds C. The data aggregators can use push or pull mechanism to refresh the data to the client.

II. LITERATURE SURVEY

Continuous Query provides advanced matching functionality, including Boolean operators and inequalities. E.g.to request all entries for which a certain field value is larger than 50 and smaller than 80.

DATA AGGREGATORS

The data refreshed through the network of data aggregators. Configured data incoherency maintained for various data items at data aggregators.

A NETWORK OF DATA AGGREGATORS

The network of data aggregators is used to refresh the results. The higher level aggregator guarantees the tighter incoherency bound as compared to the lower level aggregators. The pull and push based techniques are used to disseminate the results to the client. The data refresh at the client occurs through one or more data aggregators. The incoherency bound for data items is maintained by data aggregator. The long running queries are used for data refreshment. When these continuous queries are executed over the aggregated data items then the sources returns the result.

These queries possess the following characteristics:

As the data is changing continuously and the user is interested in the notifications these queries return the results to the clients continuously. The system with performance results using real-world traces and they have shown that their cost-based query planning leads to queries being executed using less than one-third the number of messages required than the existing scheme[1]. The results of the query can be disseminated using the client-pull-based technique so that the results of the queries over distributed data can be correctly reported, conforming to the limited incoherency acceptable to the users[2]. The authors have proposed that users can specify the acceptable incoherency as one or more of temporal, value based or updates based incoherency. They considered value based incoherency that is incoherency in the value or results of a query, as they feel that the user's requirement is more likely to be specified in value form, and further, other forms of incoherencies can be handled through much simpler approaches. In such cases, a DA needs to ensure that the incoherency in the query results. When compared to the result obtained using the precise data item values is maintained at or below the level desired by the client. Besides the bounded incoherency, another parameter of importance for the client is the fidelity delivered. 100 % fidelity implies that client's incoherency bound is never violated. Thus, with every aggregation query, the client also specifies its corresponding incoherency bound and fidelity desired[2].

An approach of a dynamic proxy caching technique which combines the benefits of both proxy-based and back end caching approaches. The authors proposed an approach for granular proxy-based caching of dynamic content[4]. They focused on the novel aspects of approach - the ability to handle dynamic content and dynamic layouts. The ability to support dynamic layouts is enabled by the Back End Monitor (BEM). The BEM resides at the back end and generates the layout for each request. This layout is passed back to the Dynamic Proxy Cache (DPC), which assembles the page that is returned to the requesting user [4]. The closeness of the client-repository assignment problem and matching problem in combinatorial optimization, they have tailored and studied two available solutions to the matching from the above discussion[5]:

MAX-FLOW MIN-COST

This model permits a client Li to obtain each data item di of interest from a different repository. As mentioned earlier, assigning these (Li,di,Cj)triples to repositories is a many- to-one weighted assignment problem, which can be solved using min-cost network flows. The cost of an assignment would ideally be a function of the communication delays, coherence requirement of the client and the repository and the load on the repository. The load on the repositories can be roughly balanced by placing a limit on the number of (Li,di,Cj) triples that are repository can serve.

STABLE MARRIAGES/ BIPARTITE GRAPH MATCHING

Consider m men and n women, and let each man and each woman rank all the members of the opposite sex by preference. The problem is to find a stable marriage between the sets of men and women, defined to be a pairing of men with women, in which there is no man m and no woman w such that m prefers w to his current partner and w prefers m to her current partner. Since we exclude pairs of men only or women only, this is equivalent to a bipartite graph matching.

The different types of examples of aggregation queries have been given. The dynamic content distribution of data aggregators is also given by the author [6]. The push approach can be used for disseminating updates the source pushes updates to its dependents in the *d*3t, which in turn push these changes to their dependents and the end-clients. Not every update needs to be pushed to a dependent only those updates necessary to maintain the coherency requirements at a dependent need to be pushed. To understand when an update should be pushed, let cp and cp denote the coherency requirements of data item x at repositories P and Q respectively suppose P serves Q. To effectively disseminate updates to its dependents, the coherency requirement at a repository should be at least as stringent as those of its dependents: cp ≤ cq.

In proposed system, the data source contains the information, the data aggregators are used to optimize and execute the client query using simulated annealing algorithm. The results are disseminated using push-based data dissemination technique[7].

III. SYSTEM OVERVIEW

Continuous queries are used to monitor the changes of the time varying data and to provide the useful results. Continuous queries from the client can be classified on the basis of results of threshold queries, entity-based queries and value-based queries. The incoherency bound is assigned to the aggregators and as per the required data items, the results are disseminated. The execution and optimization of the query take place at the data aggregators and then the results are disseminated to the client.

- ✓ In threshold queries, the user only wants whether certain are getting true or false over an aggregation of data items.
- ✓ An entity- based query finds data items that satisfy certain selection criteria.
- ✓ Value based queries, the user wants particular data item values or the result of executing some aggregation function like sum, average, min, max, ratio etc. over some subset of data items.

SYSTEM ARCHITECTURE

The system architecture is as shown in Figure 1. A network of data aggregators is present in the middle and data sources are at the top. The clients are shown at the bottom of the architecture.

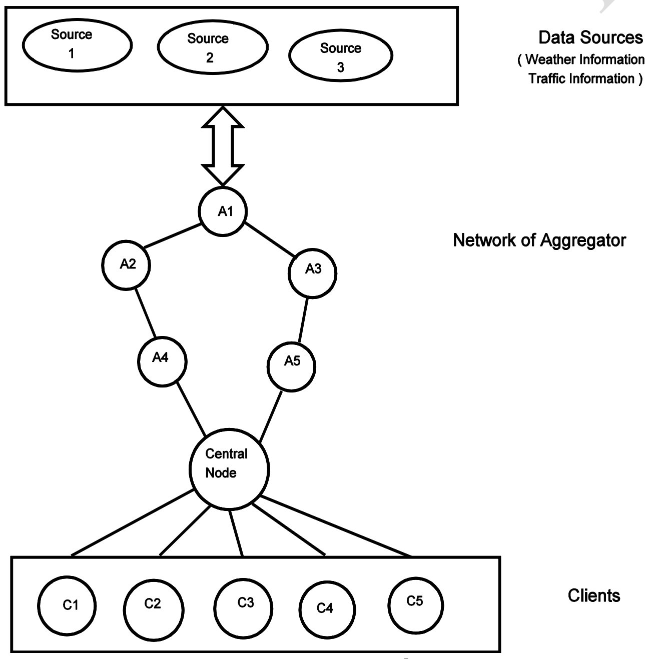

Figure 1: System Architecture

In the data dissemination network, the network of data aggregator serves the data items at specific data incoherency bound. Incoherency of the data item at a given node is defined as the difference in the value of data item at the data source and at the client. In hierarchical network of data aggregators, the aggregator at higher level guarantees tighter incoherency bound. than the data aggregators at the lower level. In the data aggregator the set (Si, Ci) present. Si is the data item and Ci is the incoherency bound of that data item. The aggregation queries MIN, MAX, COUNT, AVG, SUM are served by this technique.

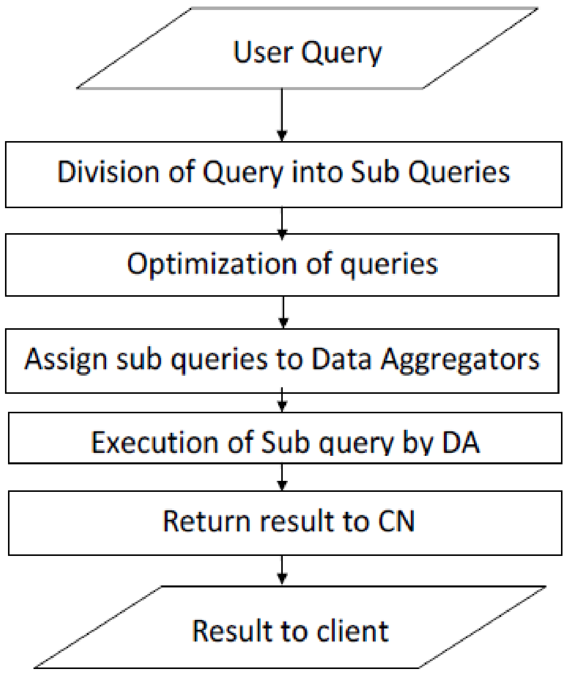
User Query
Division of Query into Sub Queries
Optimization of queries
Assign sub queries to Data Aggregators
Execution of Sub query by DA
Return result to CN
Result to client

Figure 2: System Flow Diagram

Steps for the continuous query execution:

QUERY SUBMISSION BY USER

Suppose the client wants to monitor the network data the query will be as follows:
SELECT source-department, num_packets_in_preceding_sec, FROM network_traffic_info.

This query can be used to collect the data related to the bandwidth usage of various departments. It required a continuous collection of information of data packets and their IP addresses from the routers available in the campus. A traffic policing server aggregates the information from the different departments and generates the trigger if the department crosses its allocated bandwidth. Such aggregating server uses the continuous aggregation queries to refresh the data at the particular time from specific sources. The queries considered here are the continuous multi-data incoherency bounded queries where the queries with weighted aggregation of a number of data items are considered. E.g. in the case of portfolio queries the number of shares of each stock is considered as a weight of the query. So the client query is written as:

$$V_q(t) = \sum_{i=1}^{n_q} (V_{qi}(t) * W_{qi}) \quad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where, Vq the value of the client query q involving nq data items with the weight of ith data item being Wqi, 1< i < nq.

The query results continuously need to disseminate to the client with incoherency bound Cq. Then the dissemination network ensures that:

$$\sum_{i=1}^{nq} (Sqi(t) - Uqi(t)) * Wqi| <= Cq \ldots\ldots\ldots\ldots\ldots(2)$$

Whenever the data value at data source changes then the updated value should be refreshed to the client. The data aggregator ensures that query incoherency bound Cq is satisfied:

$$\sum_{i=1}^{nq} (Cqi * Wqi) <= Cq \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

DERIVING THE SUB-QUERIES

The idea of simulated annealing comes from physical processes such as gradual cooling of molten metals, whose goal is to achieve the state of lowest possible energy.

```
Algorithm SimulatedAnnealing( )
Input: Problem, max, $t_{max}$
Output: $C_{best}$
begin
Ccurrent←CreateInitialSolution(Problem);
$C_{best}$← $C_{current}$;
for i = 1 to max do
    $C_i$ ← CreateNeighborSolution($C_{current}$);
        $t_{curr}$ ← CalculateTemperature(i, $t_{max}$);
        if Cost(Ci) ≤ Cost($C_{current}$) then
            $C_{current}$ ← Ci;
        if Cost(Ci) ≤ Cost($C_{best}$) then
            $C_{best}$ ← $C_i$;
            end
        else if Exp(( Cost($C_{current}$ )−Cost(Ci )/$t_{curr}$) >Rand() then
            $C_{current}$ ← $C_i$;
        end
    end
    return $C_{best}$;
end
```

ASSIGNING THE PARTICULAR SUB-QUERY TO THE PARTICULAR DATA AGGREGATOR WITH AN INCOHERENCY BOUND

To get the query plan the following steps are performed.

✓ Determining sub-queries.
✓ Dividing incoherency bound.

For optimal query planning the above task should be performed for following objectives.

- The sub-query qk should executable at the data aggregator ak.
- Query incoherency bound should be satisfied.
- Sub-query incoherency bound should satisfied.

✓ Execution of the sub-query by the aggregator.
✓ Dissemination of the data to the client with the help of data source:

DATA DISSEMINATION MODELS

PULL BASED APPROACH

The client need to frequently pull the data based on the dynamics of the data and a user's coherency requirement[3].

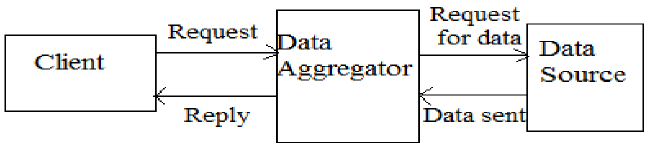

Figure 3: Pull Based Approach for Data Dissemination

PUSH BASED APPROACH

The server which has the push capability maintains state information pertaining to the client and push only those changes that are of interest to the user.

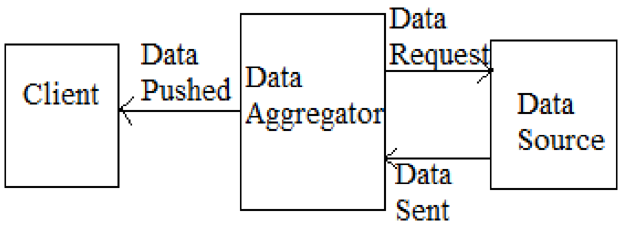

Figure 4: Push Based Approach for Data Dissemination

IV. CONCLUSION

The dynamic data is changing very rapidly. The continuous queries are used to get the updated values of this dynamic data. The continuous query is divided in sub queries and forwarded to particular data aggregator by the central node. By using simulated annealing algorithm, the optimized results can be obtained. Push based data dissemination technique is used for dissemination of the results to the client. In push based technique the user does not need to refresh data time to time. As soon as the data is available at the data aggregator the data is disseminated to the client.

REFERENCES

[1] Anindya Datta, Kaushik Dutta, Helen Thomas, Debra Vander Meer, Suresha, Krithi Ramamritham (2002), Proxy- based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. ACM SIGMOD.
[2] Elisa Bertino, Elena Ferrari, Federica Paci (2007), A system for securing push-based distribution of XML documents.
[3] S. Agrawal, K. Ramamritham and S. Shah (2004), Construction of a Temporal Coherency Preserving Dynamic Data Dissemination Network, RTSS.
[4] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham and P. Shenoy (2001), Adaptive Push-Pull: Disseminating Dynamic Web Data, 10th International World Wide Web Conference, Hong Kong,
[5] R. Gupta, A. Puri, K. Ramamritham (2005), Executing Bounded Continuous Queries at Web Data Aggregator. WWW.
[6] R. Gupta, K. Ramamritham (2007), Optimized Query Planning of Continuous Aggregation Queries in Dynamic Data Dissemination Networks, WWW.

[7] Rajeev Gupta and Krithi Ramamritham (2012), Query Planning for Continuous Aggregation Queries over a Network of Data Aggregators. IEEE 2012 Transactions on Knowledge and Data Engineering, Volume 24, Issue:6.