Data Aggregation: An Intervention Approach For Continuous Queries Using Data Aggregators

Pramod M. Bagal[1], Mr. Gajendra Singh Chandel[2]
[1]M. Tech Scholar [2]Head of Department
[1]Department of Information Technology
[1]Sri Satya Sai Institute of Science and Technology, Sehore (M.P.), India [2]Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.), India

***Abstract*—** Data aggregation is a common feature and practice where data is searched, gathered and presented in a report-based, summarized format to achieve specific business objectives or processes and/or conduct human analysis. Data aggregator is a determining factor for overall execution of continuous queries that make appropriate decisions which contribute to improved outcomes for struggling learners. The purpose of this study is to assess the information loss associated with data aggregation. Also, this study includes continuously monitoring, querying, analyzing aggregated and disaggregated data that provide results helpful for large number of users with high security, accuracy, efficiency, and scalability. Hence, we are investigates intervention approach for continuous queries using data aggregators. This assessment requires a clear methodology for implementing continuous querying against the quantifying information loss through data aggregators. The conclusion of this study is that usage of data aggregators to simplify continuous query to improve result and analyzing significant amount of information.

Keywords : data aggregator, continuous queries, data aggregation, data incoherency, incoherency bound.

I. INTRODUCTION

Data aggregation is a component of business intelligence (BI) solutions, 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 [1, 6, 10]. For such applications, data from one or more independent data sources may be aggregated continuous queries to determine if some action is warranted to improve performance. Data aggregation search databases, find relevant search query data and present data findings in a summarized format that is meaningful and useful for the end user or application.

Data aggregation generally works on big data or data marts that do not provide much information value as a whole. Data aggregation's key applications are the gathering, utilization and presentation of data that is available and present on the global Internet [2]. So there is a tremendous scalability problem here. This situation places additional burden on processing centers to extract information and produce results. As a result, users have to frequently poll the web sites of interest and fuse the newly updated information manually to keep track of changes of interest, which is a great pain.

Now days most of applications those are commercially in use having utilized the concepts of data aggregation to answer the continuous queries using data aggregators. The Web is becoming a universal medium for information publication and use [2]. It is the purpose of this study to assess the information loss associated with data aggregation. Research focusing on continuously monitoring, querying, analyzing aggregated and disaggregated data[1,6,8,9,10,11] that provide results helpful for large number of users with high security, accuracy, efficiency, and scalability.

II. BACKGROUND

A. Data Aggregation

Data aggregation is any process in which information is gathered and expressed in a summary form, for purposes such as statistical analysis. A common aggregation purpose is to get more information about particular groups based on specific variables such as age, profession, or income.

The information about such groups can be used for web site personalization to choose content and advertising likely to appeal to an individual belonging to one or more groups for which data has been collected [2]. For example, a site hat sells music CDs might advertise certain CDs based on the age of the user and the data aggregate for their age group.

B. Data Incoherency

Data accuracy can be defined in terms of incoherency of data item. It is absolute difference in value of data item at source and value of data item at client [1]. It can also be denoted as $|v_i(t)-u_i(t)|$. Where v_i= value of i^{th} data item at the source, U_i= value of i^{th} data item at the client

The data refresh message is sent to client whenever incoherency is increased exceeds C i.e. $|v_i(t)-u_i(t)|>C$.

C. Data Aggregator (DA) Network of Data Aggregators

Data aggregators can be called as organizers involved in compiling information from detailed database on individuals and selling information to others.

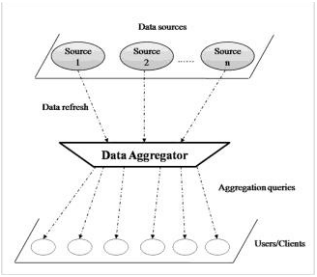

Fig. 1: Query execution at single data aggregator (DA)

Also, DA is one kind of secondary or alternate servers it serves as data sources (data items) in the computation of results from detailed database. For performing online analysis dynamic data is prominent thing. DA collects the information from designated databases and giving the data to the user.

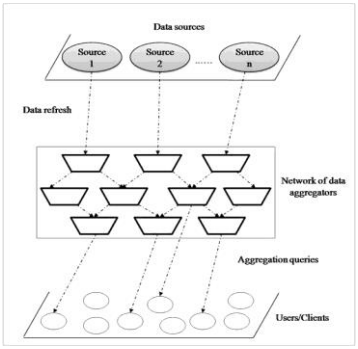

Fig. 2: Query execution at network of data aggregators

Data refresh/transfer data need to be done from source to client. The data refreshes can be done using two mechanisms [1, 6].

1) Pull Based Technique:
Source send update message to client only when client makes request.

2) Push Based Technique:
Source sends update message to client without any request that is on their own.

Fig. 2 and 3 shows data refreshes occur from data sources to clients through one or more data aggregators. 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.

D. Categorizing Queries On The Basis Of Usage

Simple Queries: Queries that can access data from single data source based on one matching condition are Simple Queries.

1) Complex Queries:
are compositions of simple queries which access data from multiple data sources and have more matching condition.

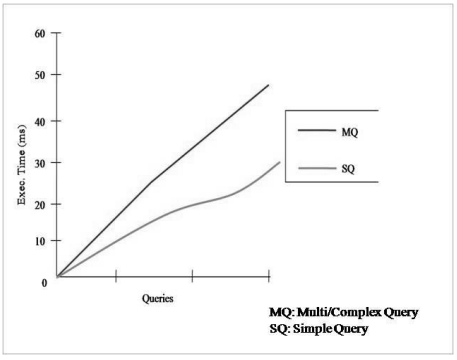

Fig. 3: Queries and its execution time evaluation

2) Continuous Queries:
Continuous queries are persistent queries that allow users to receive new results when they become available.

3) Aggregate Queries:
That possibly aggregate different sources are required to satisfy user requirement.

Fig. 3 Refer the comparison between execution time of the queries [8].

III. Literature Survey

In [1], the authors ensure that each data item for a client query is disseminated by one and only one data aggregator (refer Fig. 2). Also author prove that the problem of choosing sub queries while minimizing query execution cost is an NP-hard problem.

In [2], author the issue of aggregate diversity in recommender systems has been largely untouched. In [3], none of these work focus on the issue of storing and reutilizing parse trees. In [4], they can not address the problem of so-called concept drift and shift.

In [6], consider different pull- and push-based mechanisms to execute various types of continuous aggregation queries. In any client-server model, it's possible to refresh a data value from a server either by a client pulling the value from the server, or the server pushing it to the client. In the case of pull, the user might need to pull periodically from the server to know whether the data value has changed and to get the updated results. In the case of push, the user expresses interest in the data value to the server, and the server sends the value whenever a relevant update occurs.

In [10], involves studying two core problems: (1) how to realize meaningful and dynamic operation migration under varying system load, and (2) how to evaluate the migrated operations in the database efficiently. In this paper, prototype architecture is proposed and solution concepts for the two problems are presented.

One important question [1, 2, 3, 6] for satisfying client requests through a network of nodes is how to select the best node (s) to satisfy the request.

In [14], the authors conclude some important open problems, such as parametric query optimization, Parallelism, Routing/adaptation policies.

IV. Objectives

Executions of continuous queries have various limitations and problems are well known. Several studies have been conducted to detect and prevent execution of continuous queries. The purpose of this work is to study about execution of continuous queries using data aggregators in the form of data aggregation.

The following are the major objective:

- Allows data to be quickly, easily monitored and queried with a few keystrokes and mouse clicks.
- Eliminates additional data entry. That means no redundant data entry.
- Number of refreshes should be minimized
- Includes a fast and easy tool for generating administrative reports, queries, and listings
- Allows authorized users to design and produce highly customized ad-hoc reports and listings offers a built-in reports and listings which have proven to help improve outcomes, increase efficiency, and support best practices.
- Manage both multiple aggregators and multiple clients to delivering a better performance in terms to saves the time and the user spending low cost.

V. PROPOSE METHODOLOGY

Continuous queries are used to monitor changes to time varying data and to provide results useful for online decision making. In proposed experimental approach present a low-cost, scalable technique to answer continuous aggregation queries using dynamic data items. According to that work we employing to handle complicated query requests, such as aggregate operation. Also, as a middleware, it adds more overhead to the data querying that to minimize number of refreshes is another important contribution of this work. Compared with traditional periodic pull techniques, the experiments results are very promising.

A. Executing Queries Using Sub Queries

To answering continuous query, that required to includes the set of sub queries, their individual incoherency bounds and data aggregators which can execute these sub-queries. In experimental result, need to implement intervention approach that which satisfies client coherency requirement with the least number of refreshes.
Following 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.

VI. INTERVENTION APPROACH

A. Client-driven interventions

That protects customers from unreliable services. For example, services that miss deadlines or do not respond at all for a longer time are replaced by other more reliable services in future discovery operations.

B. Provider-driven interventions

Provider-driven interventions are desired and initiated by the service owners to shield themselves from malicious clients. For instance, requests of clients performing a denial of service attack by sending multiple requests in relatively short intervals are blocked (instead of processed) by the service.

Depending on requirements following four deployment models have been identified, each with specific characteristics that support the needs of the services and users in particular ways.

1) Security Module
This module 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.

2) Flow Chart Module
This module is used to help the user to view the value in bar flow chart based on the date. This chart to display the aggregated value based on the updating values continuously.

The companies value is changed automatically chart value is changed.

3) Update Module
This module is used to help the user to view the value to update in every minute. So the user waiting time is reduced and sees the updated value in every minute. The server is set the time when form is updated.

4) Client/Server Module
This module is used to help the client and server interaction to the database. This is possible to refresh a data value from a server either by a client pulling the value from the server, or the server pushing it to the client. In the case of pull, the user might need to pull periodically from the server to know whether the data value has changed and to get the updated results. In the case of push, the user expresses interest in the data value to the server, and the server sends the value whenever a relevant update occurs.

These values are assigning to the chart x and y position and display the client. These values are changed in dynamically based on the server entering values.

VII. CONCLUSION AND FUTURE SCOPE

In this literature presents intervention approach that answering continuous queries to allow user to receive new result when they become available. We assume the existence of DA is capable to push new content whenever they new content is available. Experimental result is very promising to minimize the number of refreshes required to execute continuous query.

In conclusion, previous work on data-driven processing and dissemination has the following problems:

- Active/dynamic queries use ad hoc table-based triggering to support dynamic data flow to the clients, which has bad scalability and performance;
- Data processing is separated from data dissemination, which leads to high overhead and lack of support for complicated query requests.

Our work can be extended in a number of directions. First this strategy reduces time taken for executing sub-queries. It is showed that dividing multi-queries into independent sub-queries and executing it parallel, improves performance and reduces execution time. Second, developing strategy for multiple invocations and changing a query plan as data dynamics changes. Third, data aggregators are dynamically assigned to sub query based on attributes so query execution plan is applicable for dataset having limited attributes. This will lead to poor performance and information loss.

Last interesting improvement is scalability of the continuous query processing. We need to ensure that it can efficiently handle large numbers of users all requesting different stocks. This can be achieved in the queue initialization phase. If the server detects too many continuous queries running on the server, it will deny some continuous query initialization requests depending on their

priority. It will enforce them to use normal query. This work gives rise to several interesting directions for future research.

REFERENCES

[1] Rajeev Gupta and Kirthi Ramamrithm, "Query Planning for Continuous Aggregation Queries Over a Network of data Aggregators", IEEE Trans. on Knowledge and Data Eng., vol. 24, no. 6, June 2012, pp. 1065-1079.
[2] Gediminas Adomavicius and YoungOk Kwon, "Improving
[3] Aggregate Recommendation Diversity Using Ranking-Based Techniques", IEEE Trans. on Knowledge and Data Eng., vol. 24, no. 5, May 2012, pp. 896-911.
[4] Luis Tari,Student, Phan Huy Tu, Jo¨rg Hakenberg, Yi Chen,Member, Tran Cao Son, Graciela Gonzalez, and Chitta Baral, "Incremental Information Extraction Using Relational Databases", IEEE Trans. on Knowledge and Data Eng., vol. 24, no. 1, January 2012, pp. 86-99.
[5] Jose Antonio Iglesias, Plamen Angelov, Agapito Ledezma, and Araceli Sanchis, "Creating Evolving User Behavior Profiles Automatically", IEEE Trans. on Knowledge and Data Eng.,vol. 24, no. 5, May 2012, pp. 854-867.
[6] Lawrence Carin, George Cybenko, and Jeff Hughes, "Cybersecurity Strategies: The QuERIES Methodology", Published by the IEEE Computer Society, August 2008, pp. 20-26.
[7] Rajeev Gupta and Krithi Ramamritham, "Scalable Execution of Continuous Aggregation Queries over Web Data", Published by the IEEE Computer Society, January/February 2012, pp. 43-50.
[8] Serge Abiteboul, T.-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart ,"Capturing Continuous Data and Answering Aggregate Queries in Probabilistic XML", ACM Transactions on Database Systems, pp. A:4-A:45.
[9] Yan-Nei Law and Carlo Zaniolo, "Improving the accuracy of continuous aggregates and mining queries on data streams under load shedding", Int. J. Business Intelligence and Data Mining, 2008, pp. 99-117.
[10] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, John Gerth, Justin Talbot, Khaled Elmeleegy and Russell Sears, "Online Aggregation and Continuous Query support in Map Reduce", SIGMOD'10, Indianapolis, Indiana, USA, June 2010.
[11] Yuanzhen Ji, "Database Support for Processing Complex Aggregate Queries over Data Streams", EDBT/ICDT '13 , Genoa, Italy, March 2013.
[12] Like Gao and X. Sean Wang, "Improving the Performance of Continuous Queries on Fast Data Streams: Time Series Case", Dept. of Information and Software Engineering, George Mason University, Fairfax, Virginia.
[13] Lory Al Moakar, Panos K.Chrysanthis, Christine Chung, Shenoda Guirguis, Alexandros Labrinidis, Panayiotis Neophytou, and Kirk Pruhs, "Auction-based Admission Control for Continuous Queries in a Multi-Tenant DSMS", *International Journal of Next-Generation Computing*, Vol. 3, No. 3, November 2012, pp.247-273.
[14] E. Pourabbas and, A. Shoshani, "Improving Estimation Accuracy Of Aggregate Queries On Data Cubes", *Data and Knowledge Engineering*, 2009, pp. 1-23.
[15] Amol Deshpand, Zachary Ives, and Vijayshankar Raman, "Adaptive Query Processing: Why, How, When, What Next?", *VLDB '07*, September 23-28, 2007, Vienna, Austria.

