International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.1, January 2013 DOI:10.15693/ijaist/2013.v2i1.21-25

Implementation of Optimization of Correlated Subqueries Execution using Parallel Way Source Definition Algorithm

S.T.Padmapriya
Post Graduate Student
Department of Computer Science and Engineering, Sethu Institute of Technology
Virudhunagar – 626115, Tamilnadu, India

Abstract - **Resource utilized group aggregation for correlated type validation at client level is the proposed access. Correlated sub query is the query to return the resultant records in partial manner. It means, the query evaluates grouped records and fetches the result until the last group is met. In large data bases there are highly maximized records to meet the groups. This type of query evaluates once and returned to result to parent and continues to fetch the retrieval process. In this project, Client Framework prepares task and it will be divided into many divisions to start separate way to start the multi connection process using parallel approach. Intelligent Group Service checks the availability of groups and decides percentage level based on the idle resources in network. Percentage tasks decide how many groups can be sent to one available server. The parallel approach starts the retrieval process from the system based on the percentage. After the downloading process, the assembler starts the assembling process. Data producer Service prepares the result and return to Client frame work. After making fast retrieval Assembler assembles it in locally. Population Data are replicated in Mirror Network. Client system finds the availability of available servers and makes the communication in dynamic level. Group based segregated process against servers is started from client process by IGS.**

Keywords: Continuous queries, Distributed query processing, Data dissemination, Coherency, Performance, Aggregation.

I.INTRODUCTION

A correlated sub query is nothing more than a sub query that is connected to the "main" query - it uses one or more columns from the main table in the SQL of the sub query. Many times, it is logically equivalent to a join; based on the current statistics, the optimizer may decide to "convert" the sub query into a join. For example, if the optimizer thinks both the query and the sub query will approximately the same number of rows, it will probably use a join. If it thinks that the sub query will return a small number (relative to the main query), it may leave it as a sub query.

The correlated sub query is a very powerful tool. It is an excellent technique to use when there is a need to determine which rows to be selected based on one or more values from another table. This is especially true when the value for comparison is based on an aggregate. It combines sub query processing and join processing into a single request. The operation for a correlated sub query differs from that of a normal sub query.

Instead of comparing the selected sub query values against all the rows in the main query, the correlated sub query works backward. It first reads a row in the main query, and then goes into the sub query to find all the rows that match the specified column value. Then, it gets the next row in the main query and retrieves all the sub query rows that match the next value in this row. This processing continues until all the qualifying rows from the main select are satisfied.

The Multithreading paradigm has become more popular as efforts to further exploit instruction level parallelism have stalled since the late-1990s. This allowed the concept of Throughput Computing to re-emerge to prominence from the more specialized field of transaction processing: Even though it is very difficult to further speed up a single thread or single program, most computer systems are actually multi-tasking among multiple threads or programs.

Techniques that would allow speed up of the overall system throughput of all tasks would be a meaningful performance gain. The two major techniques for throughput computing are multiprocessing and multithreading. If a thread gets a lot of cache misses, the other thread(s) can continue, taking advantage of the unused computing resources, which thus can lead to faster overall execution, as these resources would have been idle if only a single thread was executed. If a thread cannot use all the computing resources of the CPU (because instructions depend on each other's result), running another thread permits to not leave these idle. If several threads work on the same set of data, they can actually share their cache, leading to better cache usage or synchronization on its values.

II.EXISTING METHODS

E.Edara et.al (2008) proposed a novel technique called asynchronous in-network prediction incorporating two computationally efficient methods for in-network prediction of partial aggregate values. These values are propagated via a tree whose construction is cognizant of (a) the coherency requirements associated with the queries, (b) the remaining energy at the sensors, and (c) the communication and message processing delays. Finally, they exploited in-network filtering and in-network aggregation to reduce the energy consumption of the nodes in the network. Experimental results over real world data support their claim for aggregate queries with associated coherency requirements, a prediction based asynchronous scheme provides higher quality results for a longer amount of time than a synchronous scheme. Also, whereas aggregate dissemination techniques proposed so far for sensor networks appear to have to trade-of quality of data for energy efficiency.

Y.Zhou et.al (2008) focussed on the problem of constructing dissemination trees to minimize the average loss of fidelity of the system. They observed that existing heuristic-based approaches can only explore a limited solution space and hence may lead to sub-optimal solutions. On the contrary, they proposed an adaptive and cost-based approach. Their cost model takes into account both the processing cost and the communication cost. Furthermore, as a distributed stream processing system is vulnerable to inaccurate statistics, runtime fluctuations of data characteristics, server workloads, and network conditions, they have designed our scheme to be adaptive to these situations: an operational dissemination tree may be incrementally transformed to a more cost-effective one. Their adaptive strategy employs distributed decisions made by the distributed servers independently based on localized statistics collected by each server at runtime. For a relatively static environment, they also proposed two static tree construction algorithms relying on a priori system statistics. These static trees can also be used as initial trees in a dynamic environment.

G.Carmode et.al (2005) proposed a novel algorithmic solutions for the problem of continuously tracking a broad class of complex aggregate queries in such a distributed-streams setting. Their tracking schemes maintain approximate query answers with provable error guarantees, while simultaneously optimizing the storage space and processing time at each remote site, and the communication cost across the network. They relied on tracking general-purpose randomized sketch summaries of local streams at remote sites along with concise prediction models of local site behavior in order to produce highly communication- and space/time-efficient solutions. The result is a powerful approximate query tracking framework that readily incorporates several complex analysis queries (including distributed join and multi-join aggregates, and approximate wavelet representations), thus giving the first known low-overhead tracking solution for such queries in the distributed-streams model.

C.Ravishankar et.al (2005) proved that the client assignment problem is NP-Hard. Given the closeness of the client-repository assignment problem and the matching problem in combinatorial optimization, they have tailored and studied two available solutions to the matching problem from the literature: (i) max-flow min-cost and (ii) stable-marriages. Their empirical results using real-world dynamic data show that the presence of coherence requirements adds a new dimension to the client-repository assignment problem. An interesting result is that in update intensive situations a better fidelity can be delivered to the clients by attempting to deliver data to some of the clients at a coherence lower than what they desire. A consequence of this observation is the necessity for quick adaptation of the delivered (vs. desired) data coherence with respect to the changes in the dynamics of the system. They developed techniques for such adaptation and show their impressive performance.

S.Agrawal et.al (2004) proposed various techniques are for the efficient organization of a temporal coherency preserving dynamic data dissemination network. The network consists of sources of dynamically changing data, repositories to replicate this data, and clients. Given the temporal coherency properties of the data available at various repositories, they suggested methods to intelligently choose a repository to serve a new client request. The goal was to support as many clients as possible, from the given network. Secondly, they proposed strategies to decide what data should reside on the repositories, given the data coherency needs of the clients. They modeled the problem of selection of repositories for serving each of the clients as a linear optimization problem, and derive its objective function and constraints. In view of the complexity and infeasibility of using this solution in practical scenarios, they also suggest a heuristic solution. Experimental evaluation, using real world data, demonstrates that the fidelity achieved by clients using the heuristic algorithm is close to that achieved using linear optimization. To improve the fidelity further through better load sharing between repositories, they proposed an adaptive algorithm to adjust the resource provisions of repositories according to their recent response times. It is often advantageous to reorganize the data at the repositories according to the needs of clients. To this end, they proposed two strategies based on reducing the communication and computational overheads.

They evaluated and compared the two strategies, analytically, using the expected response time for an update at repositories, and by simulation, using the loss of fidelity at clients, as our performance measure. The results suggest that a considerable improvement in fidelity can be achieved by judicious reorganization.

C.Olston et.al (2003) proposed a new technique for reducing the overhead. 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. Their approach enables applications to trade precision for communication overhead at a fine granularity by individually adjusting the precision constraints of continuous queries over streams in a multi-query workload. Through experiments performed on synthetic data simulations and a real network monitoring implementation, they demonstrated the effectiveness of our approach in achieving low communication overhead compared with alternate approaches.

S.Shah et.al (2002), proposed techniques for disseminating dynamic data—such as stock prices and real-time weather information—from sources to a set of repositories. They focussed on the problem of maintaining coherency of dynamic data items in a network of cooperating repositories. They showed 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. However, contrary to intuition, they also showed that increasing the degree of cooperation beyond a certain point can, in fact, be detrimental to the goal of maintaining coherency at low communication and computational overheads. They presented techniques (i) to derive the “optimal” degree of cooperation among repositories, (ii) to construct an efficient dissemination tree for propagating changes from sources to cooperating repositories, and (iii) to determine when to push an update from one repository to another for coherency maintenance. They evaluated the efficacy of their techniques using real-world traces of dynamically changing data items (specifically, stock prices) and showed that careful dissemination of updates through a network of cooperating repositories can substantially lower the cost of coherency maintenance.

M.J.Franklin et.al (2002) presented the Tiny Aggregation (TAG) service for aggregation in low-power, distributed, wireless environments. TAG allows users to express simple, declarative queries and have them distributed and executed efficiently in networks of low-power, wireless sensors. They discussed various generic properties of aggregates, and show how those properties affect the performance of our in network approach. They included a performance study demonstrating the advantages of their approach over traditional centralized, out-of-network methods, and discussed a variety of optimizations for improving the performance and fault tolerance of the basic solution.

III.OVERVIEW

3.1 Description

This project describes the correlated sub query streams by means of dividing the clients query into more than one sub query. The system deals with one main server and two content servers. The main server gets request from the client and divide into many sub queries. The divided sub queries are formed by different groups. Intelligent Group Service(IGS) checks the availability and decides percentage level based on the idle resources in network. Percentage tasks decide how many groups can be sent to one available server. Parallel source definition algorithm is used for allocation of groups to the content servers. Multi threading is used to process more queries at a time, hence provide more efficient processing and time consuming.

3.2Existing System

Correlated sub query is the query to return the resultant records in partial manner. It means, the query evaluates grouped records and fetches the result until the last group is met. In large data bases there are highly maximized records to meet the groups. This type of query evaluates once and returns the result to parent and continues the retrieval process. For the simple records, the system works effectively where as large data sets suffer more process resources than the proposed system.

3.3Proposed System

Client Framework prepares task and it will be divided into many divisions to start separate way to start the multi connection process using parallel approach. Intelligent Group Service checks the availability of groups and decides percentage level based on the idle resources in network. Percentage tasks decide how many groups can be sent to one available server. The parallel approach starts the retrieval process from the system based on the percentage. After the downloading process, the assembler starts the assembling process.

Data producer Service prepares the result and return to Client frame work. After making fast retrieval Assembler assembles it in locally. Here admin server and content server are managed in network environment. Admin server is the server to enter the data in various departments. For the secured process Central Server has only storage system for population data. The proposed system design is shown in figure 3.1.

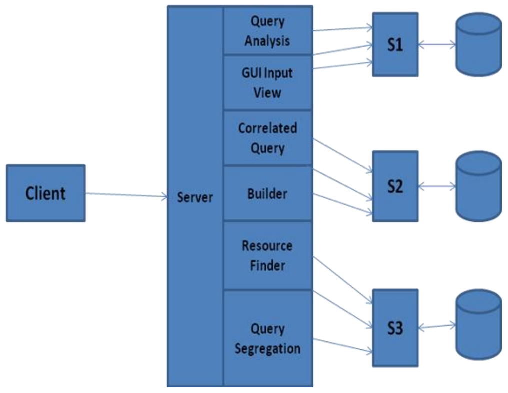

FIG 3.1 PROPOSED SYSTEM DESIGN

IV. IMPLEMENTATION AND METHODOLOGY

4.1 Parallel way Source definitions Algorithm

Step 1 : Group Counting based entity,
Step 2 : Server availability verification
Step 3 : Resource calculation Total resource calculation and it is equal to 100 Single percentage calculation Define individual percentage
Step 4 : Resource ratio based Group ratio formation
Step 5 : Parallel Connectivity

4.2 Algorithm Description

In parallel way source definitions algorithm, first the number of subqueries are counted using a CF(Counting Function). Next the availability of the server is calculated. And the percentage of the free resources in server is calculated using Intelligent Group Service(IGS). According to the ratio obtained by the IGS the subqueries are disseminated to three different servers and each subquery is assigned as a thread which enables parallel processing and thus minimize the execution time.

4.3 Modules

The modules of the project are:

1. Data constraints segment
2. Persistence mechanism
3. Query process
4. Percentage techniques
5. Partial fetching and assembling

4.3.1. Data constraints segment

The process of data constraints validates the admin accessing to start the entry level processing. The entries are stored in persistence memory using Oracle database managed in java. The System collects the necessary data to manage the person details with following types. Personal info like name, age, Official Info like job, type, experience, Tree Info like relationship with the family head, Activities Info like extracurricular activities, Education Info like graduation details, Social Info like any members in clubs or association. All the details are managed in easy GUI format with proper data constraint validation and duplicate verification. Each person will have automated generated ID known NPIN (National Person Identification Number)

4.3.2. Persistence mechanism

Oracle database contains tables to store the relevant information. The Master details are stored in master table and all child tables maintain the relationship with master table. The NPIN acts as primary and reference key to maintain the unique constraint. NPIN is the number allotted by system when the new record is added. The static records are added in tables by using defined program and staff can add the new details in all other networked systems. By using Mirror techniques, the data will be updated in different servers (3 Servers) for the data security. The updation process will be managed using parallel techniques.

4.3.3. Query process

Grouped queries are the query to group the duplicate records based given group entity. The group entity is the duplicate values column name. The CF first selects the entity column to start the group process. Now CF system counts how many grouping through the First Server. Grouping process Count can be used to define the number of way processing.

4.3.4. Percentage techniques

Dynamic percentage technique is the proposed approach in our system. Because each server has its place in network and the updated data will be stored in three different systems. When the query process is started first, IGS checks the availability of three servers. Three is not constant and it can be increased at run time, without modifying any code level process. Percentage approaches are based on system resources and percentage will be defined by runtime. For example less resource system has less data transfer task and so on. Parallel approach starts different process to the systems.

4.3.5. Partial fetching and assembling

Each one thread have the separate way and responsible to download the allotted group info from the server. No one thread interrupts the process of other threads. Because of parallel accessing, many groups data are downloaded at same time. The assembler sits at one layer and waits to complete for all threads. After the completion of all threads, the assembler starts assembling task. It means the partial data will be combined to single unit and the output process starts.

V.CONCLUSION AND FUTURE ENHANCEMENTS

Client Framework prepares task and it will be divided into many divisions to start separate way to start the multi connection process using threads. Intelligent Group Service checks the availability of groups and decides thread counts. Data producer Service prepares the result and return to Client frame work. After making fast retrieval Assembler assembles it in locally. The frontend of this Project is JAVA and the Backend is Oracle. In this project multiple threads are used.

It provides parallel communication. So that the data can be retrieved fastly from the server. It reduces the waiting time of the client. Here admin server and client are managed in network environment. Based on the groups the client apps creates the threads and the process will be decided at dynamically. After the group counting process the client apps the check the server process to make group and resource calculation. All are done in system dynamically and it is merit of our system. More type of data can be collected.

When user enters data in mobile, the data will be stored in server. To accomplish this, mobile application receives the data. By using tower communication the data will be sent to server attached Bluetooth device. Then the Bluetooth device sends the data to server. Web based environment is also possible to run under www.

REFERENCES

[1] Rajeev Gupta, and Krithi Ramamritham, "Query Planning for Continuous Aggregation Queries over a Network of Data Aggregators", IEEE transactions on Knowledge and Data Engineering June 2012.

[2] Davis, J. Parikh, and W. Weihl, "Edge Computing: Extending Enterprise Applications to the Edge of the Internet," Proc. 13th Int'l World Wide Web Conf. Alternate Track Papers & Posters (WWW),2004.

[3] D. Vander Meer, 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] 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.

[5] 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.

[6] 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.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press and McGraw-Hill 2001.

[8] 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

[9] "Query Cost Model Validation for Sensor Data,"www.cse.iitb.ac.in/~grajeev/sumdiff/RaviVijay_BTP06.pdf, 2011.

[10] 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.

Author Profile

Padmapriya.S.T received her M.sc., degree in software engineering from R.V.S. College of Engineering and Technology, India, in 2011. She is currently pursuing her M.E. in computer science and engineering from Sethu Institute of Technology, India. Her research interests include natural language processing and Query processing in web scenario.