Updates in Streaming Data Warehouses by Scalable Scheduling

Mohan Raj. A[1], M. N. Sushmitha[2]

[1, 2]School of Computing Sciences, Hindustan University
Chennai, India
mohanraj.6665@gmail.com
mnsushmitha@hindustanuniv.ac.in

Abstract: ***The project includes a streaming data warehouse update problem as a scheduling problem where jobs correspond to the process that load new data into tables and the objective is to minimize data staleness over time. The proposed scheduling framework that handles the complications encountered by a stream warehouse: view hierarchies and priorities, data consistency, inability to pre-empt updates, heterogeneity of update jobs caused by different inter arrival times and data volumes among different sources and transient overload. Update scheduling in streaming data warehouses which combine the features of traditional data warehouses and data stream systems. The need for on-line warehouse refreshment introduces several challenges in the implementation of data warehouse transformations, with respect to their Execution time and their overhead to the warehouse processes. The problem with this approach is that new data may arrive on multiple streams, but there is no mechanism for limiting the number of tables that can be updated simultaneously.***

Keywords: Data warehouse maintenance, online scheduling.

1. Introduction

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Traditional data warehouses are updated during downtimes and store layers of complex materialized views over terabytes of historical data. On the other hand, Data Stream Management Systems (DSMS) support simple analyses on recently arrived data in real time. Streaming warehouses such as Data Depot combine the features of these two systems by maintaining a unified view of current and historical data. This enables a real-time decision support for business-critical applications that receive streams of append-only data from external sources. Applications include:

- Online stock trading, where recent transactions generated by multiple stock exchanges are compared against historical trends in nearly real time to identify profit opportunities;
- Credit card or telephone fraud detection, where streams of point-of-sale transactions or call details are collected in nearly real time and compared with past customer behavior;
- Network data warehouses maintained by Internet Service Providers (ISPs), which collect various system logs and traffic summaries to monitor network performance and detect network attacks.

The goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible. Once new data are loaded, the applications and triggers defined on the warehouse can take immediate action. This allows businesses to make decisions in nearly real time, which may lead to increased profits, improved customer satisfaction, and prevention of serious problems that could develop if no action was taken.

Recent work on streaming warehouses has focused on speeding up the Extract-Transform-Load (ETL) process. There has also been work on supporting various warehouse maintenance policies, such as immediate deferred and periodic [10]. However, there has been a little work on choosing, of all the tables that are now out-of-date due to the arrival of new data, e should be updated next.

Real-time scheduling is a well-studied topic with a lengthy literature. However, problem introduces unique challenges that must be simultaneously dealt with a streaming warehouse.

Scheduling metric: Many metrics have been considered in the real-time scheduling literature. In a typical hard real-time system, jobs must be completed before their dead-lines a simple metric to understand and to prove results about. In a firm real-time system, jobs can miss their deadlines, and if they do, they are discarded. The performance metric in a firm real-time system is the fraction of jobs that meet their deadlines. However, a streaming warehouse must load all of the data that arrive therefore no updates can be discarded. In a soft real-time system, late jobs are allowed to stay in the system, and the performance metric is lateness which is the difference between the completion times of late jobs and their deadlines. However, concerned about properties of the update jobs. Instead, we will define a scheduling metric in terms of data staleness, roughly defined as the difference between the current time and the time stamp of the most recent record in a table.

2. Existing System

The traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. Where traditional data warehouse store

layers of complex materialized views over terabytes of historical data. This existing system does not support to make decisions in real time and immediately. This existing system is not suitable for data warehouse maintenance. The problem with this approach is that new data may arrive on multiple streams, but there is no mechanism for limiting the number of tables that can be updated simultaneously.

3. Proposed System

The best way to schedule updates of tables and views in order to maximize data freshness. Aside from using a different definition of staleness, our Max Benefit basic algorithm is analogous to the max-impact algorithm as is our "Sum" priority inheritance technique. Our main innovation is the Multi track Proportional algorithm for scheduling the large and heterogeneous job sets encountered by a streaming warehouse additionally; we propose an update chopping to deal with transient overload.

3.1 A Proposed System Architecture

Every time, the seller sends details about share which will be automatically streamed or updated in the top of the form before the buyer buy the particular share. The share details like company name, shares sold, available quantity etc would be updating from the database. These share details show in streaming format. The users don't need to refresh the page every time.

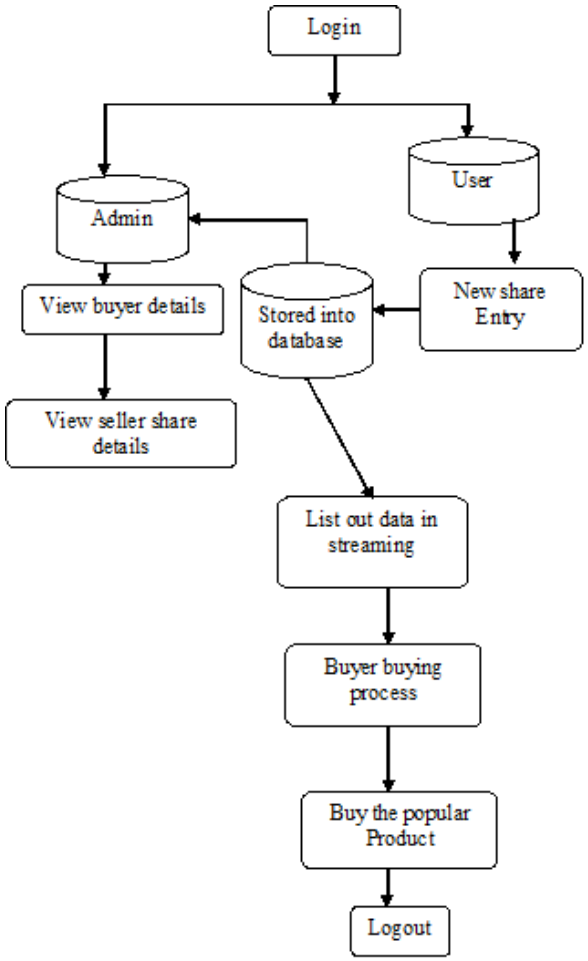

Figure 1: Proposed System Architecture

4. Literature Survey

4.1 Soft Real-Time Database System

The Proposed efficiently export a materialized view but to knowledge none have studied how to efficiently import one. To install a stream of updates, a real-time database system must process new updates in a timely fashion to keep the database fresh, but at the same time must process transactions and meet their time Constraints. Various properties of updates and views that affects this trade-off. Examining through simulation, four algorithms for scheduling transactions and installing updates in a soft real-time database [1].

4.2 Multiple View Consistency for Data Warehouse

The proposed data warehouse stores integrated information from multiple distributed data sources. In effect, the warehouse stores materialized views over the source data. The problem of ensuring data consistency at the warehouse can be divided into two components: ensuring that each view reflects a consistent stare of the base data, and ensuring that multiple views are mutually consistent. Guarantying multiple view consistency (MVC) and identify and define formally three layers of consistency for materialized views in a distributed environment [2].

4.3 Synchronizing a Database to Improve Freshness

The proposed a method to refresh a local copy of an autonomous data source to maintain he copy up-to-date. As the size of the data grows, difficult to maintain the fresh copy making it crucial to synchronize the copy electively. Two fresh Metrics, such as change models of the underlying data and synchronization policies [3].

4.4 Operator Scheduling For Memory

The proposed many applications involving continuous data streams, data arrival are busty and data rate fluctuates over time. Systems that seek to give rapid or real-time query responses in such an environment must be prepared to deal gracefully with bursts in data arrival without compromising system performance. Strategies for processing burst streams adaptive, load-aware scheduling of query operators to minimize resource consumption during times of peak load. Chain scheduling, an operator scheduling strategy for data stream systems that is near-optimal in minimizing run-time memory usage for any collection of single stream queries involving selections, projections, and foreign-key joins with stored relations. Chain scheduling also performs well for queries with sliding-window joins over multiple streams, and multiple queries of the above types [4].

5. Modules

5.1 New Share Entry

The user will upload the new share details into the database. They enter the information like id number of company, company name, date of submission of share, product code, product name, quantity, sold share, last and current year profit, and term period etc.

5.2 Seller Product Details

The company registers their product. They will enter the product code, brand name and description about the product. This is called the registration about the particular product. After feeding these data, the seller will submit on the database. When the details once are stored means, the buyer can view those details and buy the particular share.

5.3 View Share Details

The buyer can see the details regarding each share that are given by the seller. The buyer sees the product name, product code, and brand name of product etc. The data are collected from the relevant database.

5.4 List out Data in Streaming

This is the main operation between seller and buyer. Every time, the seller details about share which will be automatically streamed or updated in the top of the form before the buyer buy the particular share. The share details like company name, shares sold available quantity etc., would be updating from the database. The users don't need to refresh the page every time. These modules have to show all details about particular share in various companies. These share details show in streaming format. The users don't need to refresh the page every time

5.5 Buyer View Stock Details

This is used to view a particular product details for a buyer or a customer. Before buying the product, they can view all the information about the product. But also the data will be going streaming wise in the form more information buyer goes to view stock details page.

5.6 Buyer Buying Process

This module, the buyer gives the data to seller. The buyer gives the information like total cost of share, buyer id, buyer name, date of buying etc... And finally will submit it into the database. When completing the buying process, it will goes to streaming data in FIFO (First in First Out) method. Here if any share price and quantity will be updating means that updating share also added in streaming instead of old data's. Display the streaming data based on ranking and priorities. Here Buyer Analyze the share details history, if he satisfied with that share details means he purchase the share.

6. Conclusion

The formalized and solved the problem of no preemptively scheduling updates in a real-time streaming warehouse. The proposed the notion of average staleness as scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track. The main feature of framework is the ability to reserve resources for short jobs that often correspond to important frequently refreshed tables, while avoiding the inefficiencies associated with partitioned scheduling techniques.

7. Future Works

Future works to extend our framework with new basic algorithms. To fine-tune the Proportional algorithm in experiments, even the aggressive version with "all" allocation still exhibits signs of multiple operating domains, and therefore can likely be improved upon. Another interesting problem for future work involves choosing the right scheduling "granularity "when it is more efficient to update multiple tables together.

References

[1] B. Adelberg, H. Garcia-Molina, and B. Kao, "Applying Update Streams in a Soft Real-Time Database System," Proc.ACM SIGMOD Int'l Conf. Management of Data, pp. 245-256,1995.

[2] Y. Huge, J. Wiener, and H. Garcia-Molina, "Multiple View Consistency for Data Warehousing," Proc. IEEE 13th Int'l Conf. Data Eng. (ICDE), pp. 289-300, 1986.

[3] J. Cho and H. Garcia-Molina, "Synchronizing a Database to Improve Freshness," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp.117-128, 2000.

[4] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling Updates in a Real-Time Stream Warehouse," Proc. IEEE 25th Int'l Conf. Data Eng. (ICDE), pp. 1207-1210, 2009.

[5] B.Babcock, S.Babu, M.Datar, and R.Motwani, "Chain: Operator Scheduling for Memory Minimization in Data Stream Systems," Proc.ACM SIGMOD Int'l Conf. Management of Data, pp. 253-264, 2003.

[6] M.H.Bateni, L.Golab, M.T.Hajiaghayi, and H.Karloff, "Scheduling to Minimize Staleness and Stretch in Real-time Data Warehouses," Proc. 21st Ann. Symp. Parallelism in Algorithms and Architectures (SPAA), pp. 29-38, 2009.

[7] A. Burns, "Scheduling Hard Real-Time Systems: A Review," Software Eng. J., vol. 6, no. 3, pp. 116-128, 1991.

[8] D. Carney, U. Cetintemel, A. Raisin, S.Zdonik, M. Cherniack, and M. Stonebreaker, "Operator Scheduling in a Data Stream Man- ager," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), pp. 838-849, 2003.