Load Balancer Scheduling Over Streaming Data in Federated Databases

A. Sreeja [1] I. V. Sailaxmiharitha [2] N. Bhaskar [3]
[1]M-Tech in CSE, CMRTC, Hyderabad, India,
[2]M-Tech in CSE, CMRTC, Hyderabad, India,
[3]Associate Professor CSE, CMRTC, Hyderabad, India,

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: Online Scheduling, Data Warehouse, Data Modules, Web Database.

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.

A load balancer can be used to increase the capacity of a server farm beyond that of a single server. It can also allow the service to continue even in the face of server down time due to server failure or server maintenance. A load balancer consists of a virtual server which, in turn, consists of an IP Address and port. This virtual server is bound to a number of physical services running on the physical servers in a server farm. A client sends a request to the virtual server, which in turn selects a physical server in the server farm and directs this request to the selected physical server. Load balancers are sometimes referred to as "directors"; while originally a marketing name chosen by various companies, it also reflects the load balancer's role in managing connections between clients and servers. We then propose a 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.

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.

2. Extract Transform Load:

The term ETL which stands for extract, transform, and load is a three-stage process in database usage and **data warehousing**. It enables integration and analysis of the data stored in different databases and heterogeneous formats. After it is collected from multiple sources (extraction), the data is reformatted and cleansed for operational needs (transformation). Finally, it is loaded into a target database, data warehouse or a data mart to be analyzed. Most of numerous extraction and transformation tools also enable loading of the data into the end target. Except for data warehousing and business intelligence, ETL Tools can also be used to move data from one operational system to another.

2.1 Extraction.

The extraction step is conceptually the simplest task of all, with the goal of identifying the correct subset of source data that has to be submitted to the ETL workflow for further processing. As with the rest of the ETL process, extraction also takes place at idle times of the source system - typically at night. Practically, the task is of considerable difficulty, due to two technical constraints:

- The source must suffer minimum overhead during the extraction, since other administrative activities also take place during that period, and,
- Both for technical and political reasons, administrators are quite reluctant to accept major interventions to their system's configuration; therefore, there must be minimum interference with the software configuration at the source side.

The purpose of the extraction process is to reach to the source systems and collect the data needed for the data warehouse. Usually data is consolidated from different source systems that may use a different data organization or format so the extraction must convert the data into a format suitable for transformation processing. The complexity of the extraction process may vary and it depends on the type of source data. The extraction process also includes selection of the data as the source usually contains redundant data or data of little interest. For the ETL extraction to be successful, it requires an understanding of the data layout. A good ETL tool additionally enables storage of an intermediate version of data being extracted. This is called **"staging area"** and makes reloading raw data possible in case of further loading problem, without re-extraction. The raw data should also be backed up and archived.

2.2 Transformation.

The transform stage of an ETL process involves an application of **a series of rules or functions** to the extracted data. It includes validation of records and their rejection if they are not acceptable as well as integration part. The amount of manipulation needed for transformation process depends on the data. Good data sources will require little transformation, whereas others may require one or more transformation techniques to meet the business and technical requirements of the target database or the data warehouse. The most common processes used for transformation are conversion, clearing the duplicates, standardizing, filtering, sorting, translating and looking up or verifying if the data sources are inconsistent. A good ETL tool must enable building up of complex processes and extending a tool library so custom user's functions can be added.

2.3 Load.

The loading is the last stage of ETL process and it loads extracted and transformed data into a target repository. There are various ways in which ETL load the data. Some of them physically insert each record as a new row into the table of the target warehouse involving SQL insert statement build-in, whereas others link the extraction, transformation, and loading processes for each record from the source. The loading part is usually a **bottleneck of the whole process**. To increase efficiency with larger volumes of data we may need to skip SQL and data recovery or apply external high-performance sort that additionally improves performance.

Jobs must be completed before their deadlines 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.

3. Existing System

The closest work to ours is which finds the best way to schedule updates of tables and views in order to maximize data freshness. 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.

4. Proposed System

In this paper, we motivated, formalized, and solved the problem of nonpreemptively scheduling updates in a real-time streaming warehouse. We proposed the notion of average staleness as a scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. We 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 our 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.
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.

4.1 A Proposed System Architecture

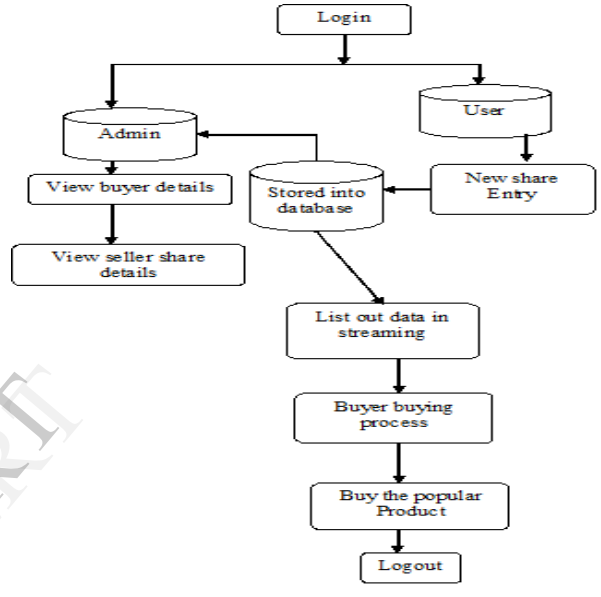

Figure 1: Proposed System Architecture

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 how in streaming format. The users don't need to refresh the page every time

.5. Literature Survey

5.1 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].

5.2 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].

5.3 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].

5.4 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].

6. Modules

6.1 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.

6.2 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.

6.3 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.

6.4 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

6.5 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.

6.6 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.

7. Conclusion

The formalized and solved the problem of no pre-emptively 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.

8. Acknowledgement

The Successful Completion of any task would be incomplete without expression of simple gratitude to the people who encouraged our work. The words are not enough to express the sense of gratitude towards everyone who directly or indirectly helped in this task.I thankful to this Organization CMR Technical Campus, which provided good facilities to accomplish my work and would like to sincerely thank to our chairman Gopal Reddy Sir, Director Dr. A. Raji Reddy Sir, Dean Dr. Purna Chandra Rao Sir, and my HOD K SrujanRaju, sir and faculty members for giving great support, valuable suggestions and guidance in every aspect of my work.

9. References

[1] B. Adel berg, 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] JAMES A.LARSON "Federated Database system for Managing Distributed, Hetrogeneous and Autonomous databases" ACM computing Surveys, Vol.22, No.3, September 1990.
[7] A. Burns, "Scheduling Hard Real-Time Systems: A Review," Software Eng. J., vol. 6, no. pp. 116- 128, 1991.