SURVEY OF STREAMING DATA WAREHOUSE UPDATE SCHEDULING

Madhuri A. Pandit[1], Rashmi Deshpande[2]

[1]M.E. 2nd Year, Department of Information Technology, Siddhant College of Engineering, Pune, Maharashtra, India
[2]Asst. Professor, Department of Information Technology, Siddhant College of Engineering, Pune, Maharashtra, India

Abstract

In this paper, we study scheduling problem of updates for the streaming data warehouses. The streaming data warehouses are the combination of traditional data warehouses and data stream systems. In this, jobs are nothing but the processes which are responsible for loading new data in the tables. Its purpose is to decrease the data staleness. In addition, it handles well, the challenges faced by the streaming warehouses like, data consistency, view hierarchies, heterogeneity found in update jobs because of dissimilar arrival times as well as size of data, preempt updates etc. The staleness of data is the scheduling metric considered here. In this, jobs are nothing but the processes which are responsible for loading new data in the tables. Its purpose is to decrease the data staleness. In addition, it handles well, the challenges faced by the streaming warehouses like, data consistency, view hierarchies, heterogeneity found in update jobs because of dissimilar arrival times as well as size of data, preempt updates etc. The staleness of data is the scheduling metric considered here.

Keywords: *partitioning strategy, scalable scheduling, data stream management system.*

----------***----------

1. INTRODUCTION

The main purpose of streaming data warehouses is to distribute new data through all related tables and views as quickly as possible. When new arriving data are loaded in the data warehouse, the triggers and applications defined over that will be able to take immediate actions. This helps to take real time decisions to increase profit in business. It also avoids any serious problem and improves customer satisfaction. Recent work on streaming data warehouses is mostly related to ETL (Extraction, Transformation, Load) process. But there has been less work on selection of all the obsolete tables, which have become obsolete because of appearance of up-to-the-minute data. The new data may emerge on several streams, but there is no way for limiting the number of tables that can be updated at the same time. Hence there is a need of scheduler which limits the number of simultaneous updates and schedules the next job. Real time scheduling problem is well studied by means of lengthy literature. The challenges generally found with scheduling like non-preemptibility, hierarchies, scheduling metrics, data consistency, priorities, transient load, and heterogeneity and are concurrently handled by the streaming data warehouses [1].

We study the paper, which demonstrates how to schedule updates for views in the form of streams and the transactions in the real time database system. In this, two definitions for staleness are given. First is MA (Maximum Age) and second is UU (Unapplied Update). Four algorithms are used for the scheduling transactions and installing the updates in the soft real time database systems. We study the paper in which the scheduling strategy affects the performance metrics like, tuple latency, throughput, and memory necessities for the systems in which query processing is continuous. Another paper gives focus on the problem of scheduling the updates, when there are materialized views. Then it finds the best order to update them to exploit the quality of data (QoD) in presence of continual updates. The group-EDF (gEDF) algorithm uses the dynamic grouping of tasks by deadlines. Deadlines of tasks are very close to each other. SJF (Shortest Job First) algorithm is used to schedule the tasks inside a group. Thus total execution time is minimized.

Scheduler is the key component of real time systems which assigns inadequate resources to serve request in due course. The further constituent of real time database systems is organization of input streams and applying update for the corresponding table in the database. Example of such input streams are data from sensors in engineering control systems, service request for telecommunication systems, call request or return the state of supplementary databases in system. The main intend is to maintain the external data consistency. If the size of materialized views is restricted, the number of updates per second is so short that the resource requirements for views are definitely trivial [2].

In DSDM, scheduling problem is very complex. It has substantial effect on the performance metrics like tuple latency, system throughput. The resources like CPU, I/O and main memory are fixed, but the scheduling jobs can be extremely dynamic. Predefined QoS requirements for query put many constraints. The efficient strategy for scheduling in DSDM must be able to: (1) achieve the maximum performance with the fixed amount of resources, (2) guarantee the specified QoS for the query if required, (3) take suitable actions under unanticipated conditions, (4) strategy must be implemented easily and run effectively when there is small overhead. The actual strategy possibly

will not be able to accomplish all the above properties, since there are tradeoffs amongst all the performance metrics and usage of the offered resources. The path capacity strategy gains best tuple latency. The segment scheduling strategy gives the least memory requirement. The threshold strategy provides sound overall performance, despite the fact that they do not meet all the properties [3].

To maximize the overall QoD (quality of data), a set of materialized views are set to the best order to update the views. FIFO schedule for the web views updates can have catastrophic effect on QoD. QoD-aware update scheduling algorithm (QoDA) combines the scheduling of tables and view updates in single framework. QoDA scheduling can keep a high level of QoD. Still the update processing capacity is not sufficient and there are surges in the arriving rate. QoD utilizes temporal locality in arriving update stream, moreover considers the database schema and also supports all types of views and view hierarchies. In testing, QoDA update schedules always outperforms FIFO schedule by up to two orders of magnitudes. Also FIFO never restores QoD or it may restore it gradually. Whereas QoDA quickly restores the QoD [4].

The complexity of scheduling of the data-loading jobs is considered in order to minimize the staleness of the real time streaming data warehouses. The weighted staleness and the stretch can be restricted under some conditions on the process speed and arrival time of emerging data. In the applications like IP network monitoring, online financial trading, credit card fraud detection, the data warehouse collects the great number of streams of data feeds which are generated by the external sources. For the different tables, data are generated at different rates, but for each table, it is generated at constant rate. When the new data arrives, the triggers defined over it release an update job that appends the new data to the associated table. When the processors are adequately fast, the constant-stretch algorithm for the quasiperiodic model is used, in which the tables can be clustered in a small number of groups. The update frequencies within each group change at the most a constant factor [5].

Earliest Deadline First (EDF) scheduling algorithm is the first dynamic priority-driven scheduling algorithm. It uses priority for scheduling real time jobs. These priorities are given statically or dynamically by the system. EDF suffers considerably if the system is overloaded. EDF can be online or offline, based upon the selection of the types of jobs involved. Generally, offline scheduling has great performance than online scheduling. But it may lead to poorer utilization of resources. The metric of the real-time systems is the success ratio of the system deadlines. Success ratio is the percentage of jobs finished before their deadlines. Another metric like the minimized total SJF scheduling is optimal, but it requires expected execution time to be completely applied. SJF algorithm can be applied either non-preemptively or preemptively. SJF has short average waiting time. It is optimal regarding average waiting time. This approach is superior compared to other algorithms [8].

2. STREAMING DATA WAREHOUSE

2.1 Streaming Data Warehouse Architecture

The architecture of the streaming data warehouse is shown in Figure 1. It has two types of tables, base tables and derived tables. Initially, the base tables are loaded from the data streams. Then the derived tables are defined as an SQL query over those base tables as the materialized view. Each table has user defined priority Pn as well as time-dependent staleness function Sn(t). A dependency graph is maintained which indicates the relationship between the source and derived tables. This graph can be acyclic and directed. For each table Tn, we have, (1) the set of ancestors, and (2) the set of dependent tables. Ancestors directly or indirectly serve as its sources, whereas dependent tables are directly or indirectly sourced from it. On the arrival of new data in the form of stream, an update job Jn is released. This job executes the ETL process and loads the new data into related table Tn. Also updates its indices. After execution of update job for base table Tn, update jobs for all dependent tables are created. As soon as those jobs are finished, update jobs for other remaining tables are released in the BFS order given in the view dependency graph. Which job is going to be executed next is decided by the scheduler. Each update job is the atomic task.

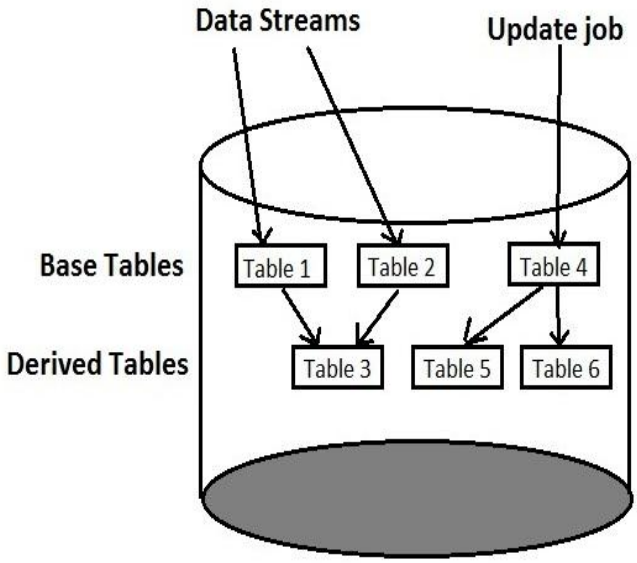

Fig - 1 : Architecture of Streaming Data Warehouse

2.2 Data Staleness

Staleness of the table Sn(t) at time t , is the difference between t and the freshness of Tn. Staleness starts increasing as soon as update is done. Priority given to the table Tn is Pn. Its objective is to minimize the total priority-weighted staleness(S) over time.

$$S = \Sigma\, Pn \int Sn(t)\, dt$$

Main aim is to minimize priority-weighted data staleness [1].

2.3 Scheduling Model

Let Ji be the new update job for the table Ti. For base table, Ji is the period of its source stream and for derived table; Ji is the maximum period of any of the Ti's ancestors. Δ Fi, i.e., freshness of Ti is defined as an increase in freshness after Ji is finished. Let n is time interval of the data which is to be loaded. Then the execution time Ei is given as,

$$Ei(n)=\alpha i +\beta i*n$$

Where,
αi = Time to initialize the ETL process
βi = Data Arrival Rate

When set of new data emerges, a new update job is released. Multiple update jobs may be pending for the same table, if the warehouse is busy executing another job. All such instances of pending update jobs are combined together into the single update job. This job loads all the available data into corresponding tables. This approach is more efficient than executing each such job independently. Because in this approach, we need to pay the fixed cost αi only once.

3. SCHEDULING OF UPDATES

In update scheduling problem, the schedule of updates is found which maximizes the quality of data. Consider the database which contains n relations and m views. Let r1, r2, r3..., rn are the relations and v1, v2, v3, v4....., vm are their views.

DAG (Directed Acyclic Graph) and View Dependency Graph is used to represent derivative paths of views. The nodes of view dependency graphs indicate either relations or views. If node b is derived from node a, then there is an edge between node a and node b. Suppose we know the cost to update each relation and the cost to refresh each materialized view. The actual update cost is not required. But their relative update cost is required. Figure 2 shows the view dependency graph. For the simplicity, consider that all update and refresh operations require one time unit excluding views v2 and v3. Finally there are only two updates, for relations r1 and r2. Update for r1 arrives at time 0, while update for r2 arrives at time 3.

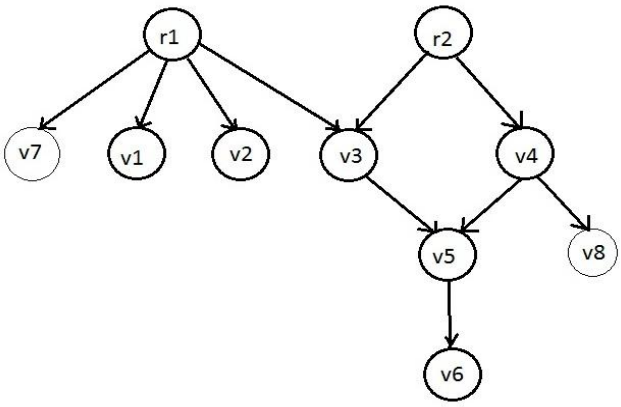

Fig -2 : View Dependency Graph

In case of FIFO update propagation schedule, the refresh of all the affected views is performed after the update to the parent relation is finished. If there are multiple paths for the single view, unnecessary refreshes must be avoided while scheduling. In figure 2, as soon as an update job for r2 has arrived, we would use schedule r2-v3-v4-v5 instead of r2-v3-v5-v4-v5 for avoiding refresh of view v5 twice. Using BFS (Breadth First Search) technique on view dependency graph, FIFO update scheduling avoids unnecessary refreshes.

3.1 Scheduling Algorithm

Basic algorithm prioritizes jobs so that that jobs can be executed on separate tracks, EDF(Earliest deadline First) algorithm orders jobs by proximity to their deadlines, since jobs are having priority EDF is not best solution because it's performance is poor. Instead EDF One of the Basic algorithm such as Prioritized EDF, Max benefit, Max benefit with lookahead can be choosen[1].Four scheduling algorithms are there for addressing the two closely related components:Do Update First(UF), DoTransaction first(TF), Split Updates(SU), Apply update On Demand (OD) [2]. FIFO strategy and Chain strategy are used to show impact of strategy on tuple latency, throughput of query processing system and queue size[3] QoD-Aware update scheduling Algorithm unifies scheduling of relation updates amd view refreshes under single framework. QoDA algorithm maintains a set of stale database objects, and at each step it selects the object with maximum impact value. QoDA algorithm can be very fast also it adds very little overhead to the system, as it has no time-dependent computation. QoDA schedule quickly recovers and maintains high quality of data [4].

3.2 Job Partitioning

If a set of jobs is heterogeneous with regard to execution time and period, scheduler performance is probably beneficial, if it is guaranteed that some part of the processing devices will be allocated to small jobs. Global scheduling is the best way to achieve better results in a soft real time setting. There are two methods for ensuring that resources will be allocated to small jobs: (1) EDF-Partitioning, (2) Proportional partitioning. EDF partitioning algorithm assigns jobs to the tracks. The deadline of an update job is equal to the release time of the job plus its period. EDF strategy is compatible with any local scheduling algorithm for scheduling on the individual track. This strategy promotes short jobs to an idle track which already contains some long jobs. There is negligible computational overhead in case of EDF algorithm. Proportional partitioning strategy forms the clusters of similar jobs. It can assign job cluster to any number of tracks. The overhead of the proportional algorithm is small. In worst case, it varies with the availability of tracks.

3.3 Hierarchy of Views

Since views are dependent on other views, the prioritization of the jobs becomes hard. It is necessary that, source view

inherits the priority of its child view. There are three ways of inheriting the priority. Those are Sum, Max and Max-plus.

3.4 EDF Scheduling with Priorities

EDF scheduling algorithm is implemented as dynamic and priority driven algorithm. In the real time systems, the priority levels should not exceed 32. EDF works on the principle of scheduling of the jobs based on the deadlines. Therefore maintaining one base priority others can be made dynamic priorities. The deadline of any job finds its priority between all the jobs having the same level of base priority.

4. COMPARATIVE ANALYSIS

The update scheduling in streaming data warehouses is discussed in large extent. Various metrics are used consistently to illustrate the best schedule. The notion of collaborating algorithm presented in paper 1 and 8 can improve the performance of update scheduling in the streaming data warehouses. This idea can escort us to the new area of research. We can choose the best order for refreshing the materialized views to improve the quality of data.

There are numerous basic algorithms for the scheduling of the updates in streaming data warehouses. Global partitioning is used to scale the database. There are two types partitioning algorithms, EDF partitioning and proportional partitioning. EDF partitioning algorithm with priorities is used for scheduling the jobs. Proportional partitioning identifies the clusters of similar jobs. Then it allocates these clusters to any number of tracks. Previous approaches have used only either EDF partitioning strategy or proportional partitioning strategy.

5. RESEARCH DIRECTIONS

In future a framework can be designed to handle complex environment of a streaming data warehouse. Previous strategies have used a single strategy, either EDF partitioning or proportional strategy. We can try to combine best features of both strategies to make scheduling of updates even faster than previous approaches. Also we can use different scheduling algorithms at different stages of scheduling.

In current systems, streaming fetches information only from one table at a time. With further research can be solved in order to fetch data from multiple tables at a time. The new data possibly will emerge on multiple streams. There is no way for limiting the number of tables that can be concurrently updated. Hence the need for a component occurs which will be able to limit the number of concurrent update jobs and will decide which job to schedule next.

6. CONCLUSION

The problem of scheduling updates in streaming data warehouse is studied. Updating streaming data warehouse is a problem in which jobs correspond to the processes. Purpose of these processes is to refresh the tables in the data warehouse. And target is to improve data freshness. The decision about which job to schedule next depends on effect of that updating on data staleness. Average staleness was measured as scheduling metric. And the algorithms were built to deal with the environment of the streaming data warehouse. The update jobs are non-preemptive. The idea of average staleness as a scheduling metric is used. It helps to take real time decisions for business critical applications. It is used in many applications like stock exchanges, online analysis of stock prices, network analysis, monitoring applications etc.

ACKNOWLEDGEMENTS

The authors would like to thank M.E. co-coordinator Prof. Mrs. Rangdale and other staff of I.T. department of Siddhant College of Engineering, Pune for their helpful discussions.

REFERENCES

[1]. Lukasz Golab, Theodore Johnson, and Vladislav Shkapenyuk,”“Scalable Scheduling of Updates in Streaming Data Warehouses”,”IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 6, JUNE 2012”

[2]. 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.

[3]. Qingchun Jiang, Sharma Chakravarthy,” Scheduling Strategies for a Data Stream Management System”.

[4]. Alexandros Labrinidis, Nick Roussopoulos,” Update Propogation Strategies for Improving the Quality of Data on the Web”,proceeding of the 27th VLDB Conference , Roma, Italy, 2001.

[5]. 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.

[6]. L. Golab, T. Johnson, J.S. Seidel, and V. Shkapenyuk,
“Stream Warehousing with Datadepot,” Proc. 35th ACM SIGMOD Int’l Conf. Management of Data, pp. 847-854, 2009.

[7]. 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.

[8]. Li, Wenming, “Group-EDF - a new approach and an efficient non-preemptive algorithm for soft real-time systems”. Doctor of Philosophy (Computer Science), August 2006, 123 pp., 6 tables, 49 illustrations, references, 48 titles.

BIOGRAPHIES

Madhuri Pandit is completing Master's degree in Engineering in Information Technology from Siddhant College of Engineering, affiliated to Savitribai Phule Pune University. She received Bachelor's degree in Engineering in Computer Science from Gharda Institute of Technology, affiliated to Mumbai University.

Rashmi Deshpande is assistant lecturer at Siddhant College of Engineering, affiliated to Savitribai Phule Pune University. She completed her Master's degree in Engineering in Electronics from Rajarshi Shahu College of Engineering, affiliated to Savitribai Phule Pune University. She received Bachelor's degree in Engineering in Electronics and Telecommunication from Usmanabad College of Engineering, affiliated to Dr. Babasaheb Ambedkar University, Aurangabad.