Multitrack Algorithm for Updating Streaming Data Warehouses

Miss Madhuri Ashok Pandit
M. E. 2nd Year, Department of Information Technology
Siddhant College of Engineering, Sudumbare
Pune, India
e-mail: madhuripandit15@gmail.com

Prof. Rashmi Deshpande
Asst. Professor, Department of Information Technology
Siddhant College of Engineering, Sudumbare
Pune, India
e-mail: rashmi2810@gmail.com

***Abstract*—The scheduling problem for streaming data warehouses for update jobs is handled. In the proposed algorithm, jobs are the updates for tables as well as views. The purpose of these jobs is to refresh tables. Main goal is to minimize the data staleness over time. For the scaling purpose, different partitioning strategies have been used. The best features of EDF partitioning and proportional partitioning are used in this algorithm. Track (Logical partition of resource) utilization is calculated. Track utilization is directly proportional to the speed of the data warehouse update process.**

Keywords- Proportional; Partitioning; Scheduling Algorithm; Streaming data warehouses; Track Utilization

I. INTRODUCTION

The data warehouses these days are experiencing tremendous growth. That's because it is the dominant decision support tool .Traditional data warehouses are refreshed in down times. They contain layers of composite materialized views above terabytes of historical data [1]. Data Stream Management Systems (DSMS) hold up simple analysis on newly appeared data in real time. For the purpose of improving efficiency of the EDF algorithm in overloaded state as well as for under loaded state dynamic grouping of the tasks is made. Deadlines of those are close to each other. Then Shortest Job First method is used for sorting the jobs inside any group [4]. The applications having business critical requirements get append-only streams from the external sources.

Recent work on the streaming warehouses focuses on increasing the speed of ETL (Extraction, Transformation, Load) process. Also it focuses on the maintenance policies like updating the views every time the base table changes and updating the view simply when it is queried. There has been several works on finding the obsolete tables which are the result of arrival of new data, and arranging such tables for update next.

The main purpose of streaming data warehouse is to propagate new data across all the related tables and views as quick as possible. As soon as new data is loaded in data warehouse, the triggers and applications defined on them can take immediate actions. This helps us to make immediate decisions for business. This leads to increase in profit, avoids serious problems and improves customer satisfaction. The new data may emerge on multiple streams. There is no method for limiting the number of tables that can be concurrently updated. Hence the need for scheduler occur which limits the number of concurrent updates and decides which job to schedule next. The challenges in scheduling such as scheduling metrics, data consistency, hierarchies and priorities, heterogeneity and non preemptibility, transient load are simultaneously handled in streaming data warehouse [1].

Scheduler is the main part of the real time database systems. It has to allocate inadequate resources to complete the service request in time. The another component in real time database system is managing the input streams of data and applying the corresponding update for tables in the database. In this method, an algorithm orders jobs by some logical means. This algorithm can be extended to handle the complications encountered by the streaming data warehouses. To scale the large and diverse job sets, this system combines the features of global scheduling with the some features of partitioned strategy. In partitioned scheduling, we combine the update jobs by their execution time. Each cluster defines a partition and runs its own local algorithm. Generally, at most one job from each partition can run at any given time. Whereas under some conditions, we allow some pending jobs to be scheduled on a different tracks, if their "home" track is busy. So we can reserve processing resources for short jobs while achieving varying degrees of global scheduling.

Applications includes: 1. Online stock trading 2. Monitoring applications 3. Network traffic analysis 4. Financial tickers 5. Transactional log analysis.[1]

II. RELATED WORK

Updating streaming data warehouse is a problem is 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 [1]. The projected system is the algorithm which uses a two-level strategy for scheduling non-pre-emptive jobs. Tasks are clustered dynamically and shortest job first (SJF) algorithm is applied for scheduling the jobs inside the groups [4]. To keep the database up-to-date, the real time database system should process the new arriving updates in timely manner. At the same time, the transactions should be processed considering their time constraints. Four algorithms were designed for scheduling the transactions and processing the updates [7]. Data warehouse is divided in two parts for ensuring data consistency: for ensuring each view shows a consistent base table, ensuring that the views are consistent

mutually. Once it has made sure that multiple views are consistent, identify and make three layers of consistency for the materialized views in the distributed system [8]. A technique to refresh to a local copy of independent data source to keep the copy fresh is given. With increase in data size, it becomes hard to keep the up-to-date copy, making it critical to make the copy synchronized. By changing the model of the underlying data, synchronization policies as two metrics are used [2]. In an operator strategy for scheduling, the chain scheduling for the data stream systems is used, that is nearly optimal in the runtime memory usage for any set single stream queries which involves the projections, selections, foreign-key joins with stored relations. The chain scheduling can be used for the queries with the sliding window joins over numerous streams [5]. When the database and the set of relations which are materialized, and also the incoming relation update stream is given, the schedule for update that maximizes the overall quality of data.

Consider the database with two relations r1, r2 and eight materialized views v1 to v8. Views v1 to v6 are materialized views. Also suppose all updates and refresh jobs take 1 time unit for the views except for view v2 and v3. We have only two updates, one for r1 at time 0 and for relation r2 which arrives at time 3. By using FIFO update propagation strategy, all the affected views can be updated as soon as the update for base relation is done. The FIFO update schedule is better for avoiding unnecessary updates. FIFO uses breath first search (BFS) traversal of the view dependency graph to calculate the refresh order [6].

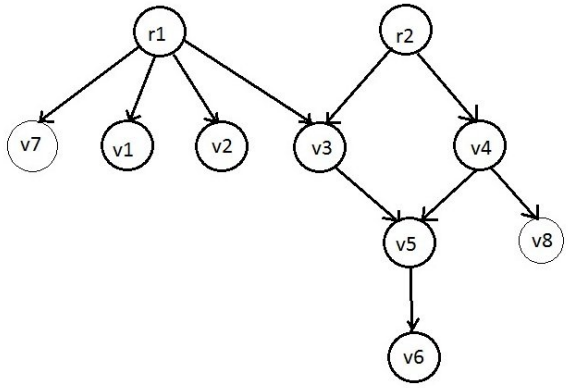

Figure 1. View Dependency Graph

III. SYSTEM MODEL

A. *Streaming Data Warehouse*

Two types of tables are maintained in streaming data warehouse, base table and derived table. These tables are stored on hard disks either partially or fully. The base table is loaded from the data stream, while the derived table is the view defined on one or multiple base or derived tables. The base and derived table Tj has user defined priority Pj and time dependent function Sj(t). Relationship between base table and derived table can be viewed by acyclic and directed dependency graph.[1] Each data stream i is generated by external sources. One or more records are passed to the data warehouse with period Pi. If the period of a stream is unknown, then the user has freedom to choose the period with which the new data should be checked by the warehouse. When new data arrives on stream i, an update job Ji is released. Ji is inserted into the scheduler queue. Task of this job is to execute ETL process, load new data in to the corresponding base table Ti, and update indices. When update for base table is completed, update jobs are released for derived tables of that source base table. The aim of the update scheduler is to choose which of the released update job to execute next. If more than one updates for a given table are pending, then they must be done in sequential manner according to their arrival time.

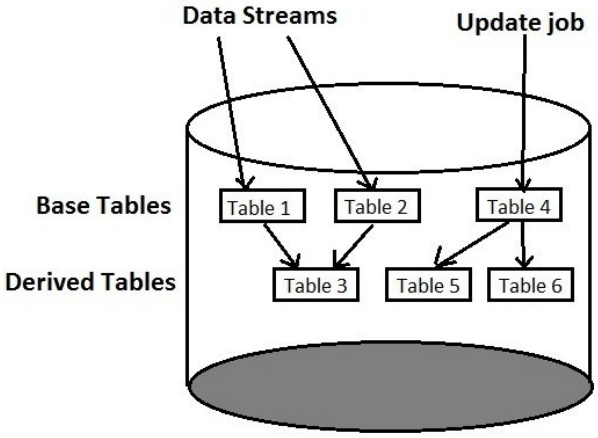

Figure 2. Streaming Data Warehouse

B. *System Architecture*

This paper presents the model as shown in the Figure 3. When the updates for the table appear on stream, it is transformed into jobs. Using PEDF algorithm, these jobs are sorted along with their arrival time. This list of sorted jobs is then divided into suitable clusters dependent on some criteria.(such as jobs having arrival time between 1-5 is assigned to first cluster and likewise). Jobs within one cluster are sorted by SJF (Shortest job first) algorithm by their execution time and average data.

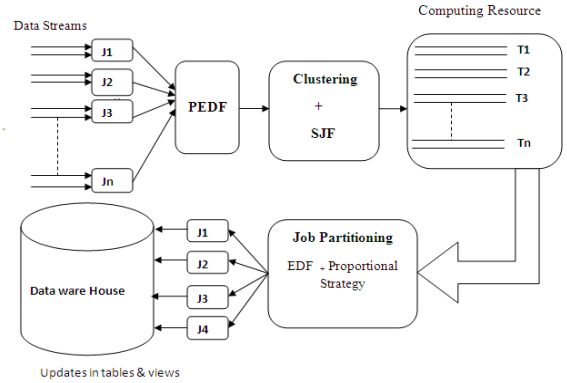

Figure 3. System Architecture

The computing resource is then logically divided into multiple tacks. Traditional way to ensure resource allocation is job partitioning and then scheduling the partition separately. The recent results indicate that better performance is achieved by global scheduling in real time environment. Two methods were investigated for ensuring resources: EDF Partitioning strategy and proportional partitioning strategy [1].

Our algorithm proves that the performance is better when some features of these two strategies is combined, than using only EDF partitioning Strategy. When all the jobs are scheduled & executed by this new algorithm the data warehouse updates are done concurrently.

IV. SCHEDULING MODEL

Consider table Ti and the update job is Ji. For the base table, the period of Ji and the period of its source stream are the same. The freshness of table is defined as an increase in the freshness after Ji is executed. Let n is the time interval of data to be loaded. Execution time of the update job,

$$Ji = \alpha i + \beta i * n$$

Where, αi = Time for initialization of ETL process and ßi = Rate at which data arrives. If there are no any updates for the same table, then all the updates are combined into single job which loads all the data into the table.

A. Scheduling Algorithm

Update jobs are partitioned considering expected time required for processing. The computing resources are divided into tracks. Any released update job is placed into queue of particular partition. Scheduler is responsible for making scheduling decisions. Local algorithm on individual tracks prioritizes their jobs. Our scheduling algorithm is given as follows:

1. Create the jobs from updates emerging in stream format.
2. Sort the jobs by using PEDF algorithm.
3. Create multiple clusters and assign jobs to different clusters with respect to some criterion.
4. Sort the jobs contained by the cluster by SJF algorithm and give priorities to jobs.
5. Split the computing resources in logical tracks.
6. Partition the jobs using EDF and proportional partitioning strategy.
7. Update the data ware house.

Terminologies used for the scheduling algorithm:

- Emax = Execution time of largest job i.e. maximum processing time of job inside a track
- Pmin = Minimum period of the job inside the track
- U= Track utilization
- Total track utilization =Σ iUi

Using this algorithm we update the data warehouse quicker than the other methods. We use some of the features of EDF and Proportional Partitioning strategy together in our algorithm.

EDF Partitioning algorithm is as follows:

1. Sort released jobs by using the local algorithm.
2. For each job Ji in the sorted order
    a. If Ji's home track is vacant, schedule Ji on that home track.
    b. Else, if there is another available free track, schedule Ji on that free track.
    c. Else, look into the tracks r, such that Ji can be promoted to track r
        i. If track r is available and there is no released job remaining in the sorted list for which r is the home track,
            A. Schedule Ji on track r.
3. Else, delay the execution of Ji.

Proportional Partitioning strategy is as follows:

1. Arrange the jobs with increasing execution time
2. Create an initial cluster C0.
3. For each job Ji, in order
    a. If Ei(Pi) is less than k times the minimum period in the current cluster
        i. Include Ji in the current cluster.
    b. Else, create the new cluster, make it the current cluster, and add Ji to that cluster.
4. For each cluster Cj
    a. Compute UCj =Σ Ei(Pi)/Pi, ranging over tasks in Cj
5. Compute total utilization = ΣCj.

V. EXPECTED RESULT

The updates arrived are converted into jobs. Those jobs are then sorted with respect to their arrival time using PEDF algorithm. Then this list of sorted jobs is then clustered into four groups. Job which fullfill the criteria of the cluster is included into that cluster. Then the SJF algorithm is used to sort the jobs inside the cluster. So the jobs having minimum average data are located at the top position. The multitrack algorithm makes use of two partitioning strategies, EDFand Proportional. Jobs are allocated to different tracks accordingly. We have taken four tracks. After that we calculate the values for Emax and Pmin as well as the track utilization by using multitrack algorithm. Also we calculate track utilization by using EDF partitioning algorithm. After comparing both results, multitrack algorithm should give better result than EDF partitioning algorithm**.**

The screen shown below is the initial window. It has three buttons, Load, View and Priority EDF. Load button

is to load the data into the data warehouse initially. View button is to view the information about that data like arrival time, execution time etc. on the screen and Priority EDF button is to sort the data according to their arrival time.

Figure 4. Snapshot of Initial Window

VI. CONCLUSION

The problem of scheduling updates in streaming data warehouse is formalized and solved. These update jobs are non-preemptive. The idea of average staleness as a scheduling metric is used. Also I presented the scheduling algorithm designed to handle complex environment of a streaming data. The framework uses advantageous features of two different partitioning strategies, unlike the previous approaches which used only one strategy. And it uses two different scheduling algorithms at different stages of scheduling. 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.

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" J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

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

[3] 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.

[4] 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

[5] B. Babcock, S. Babu, M. Datar, and R. Motwani, "Chain:Operator Scheduling for Memory Minimization in DataStream Systems," Proc. ACM SIGMOD Int'l Conf.Management of Data, pp. 253-264, 2003.tables, 49 illustrations, references, 48 titles.

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

[7] 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. R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

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