Survey Study on Scalable Scheduling of Updates in Streaming Datawarehouses

[1] Prof. Pawar Bharat M. Lecturer, Computer Engineering Department, RSIET Pedhambe Chiplun, Maharashtra, India
[2] Prof. Jadhav C.M. HOD, Computer Science Enginnering Department, BIGCE Solapur, Maharashtra, India

ABSTRACT

This survey deals with the update scheduling problem in streaming data warehouse, The proposed multitrack algorithm solves the problem of updating streaming data warehouse by scheduling the jobs where jobs corresponds to updates for tables as well as views. Those jobs load new data into tables, aim is to minimize the data staleness over time. For scaling purpose we use partitioning strategies. The partitioning strategies such as EDF partitioning and some part of proportional partitioning, are combined in multitrack algorithm. We calculate track utilization, if track utilization is higher then updates are done faster in data warehouse. Another paper focuses on the update scheduling problem, when the views are materialized, find the best order to refresh them to maximize the quality of data (QoD) in presence of incessant updates. The group-EDF (gEDF) algorithm is based on dynamic grouping of tasks with deadlines. Deadlines are very close to each other, SJF algorithm is used to schedule task within a group. Hence total execution time is minimized. Using Round Robin each job is assigned a time interval, called its quantum, which is allowed to run. If the job is still running at the end of quantum, the CPU is preempted and given to another job. If the job has blocked or finished before the quantum has elapsed, the CPU switching is done. Of course, round robin is easy to implement.

Keyword:-*Scalable scheduling, Data Stream management system, streaming data warehouse, QoD (Quality of Data), Strategy.*

1. INTRODUCTION

The aim of streaming data ware house is to disseminate new data across all the related tables and views as fast as possible. When new data are loaded in data warehouse the triggers and
applications defined on data warehouse can take instant action. This leads to make real time decisions for business, hence profit increases also prevents serious problems and improves clientele satisfaction. Recent work on streaming data warehouse related to ETL process however there has been less work on selecting of all the tables that are out-of-date due to arrival of up-to-the-minute data. The new data may appear on multiple streams, but there is no method for restraining the number of tables that can be simultaneously updated Hence the need for scheduler occur that limit the number of concurrent updates and determine which job to schedule next. Scheduling problem in real time is well studied with lengthy literature. 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 heart of real time database system that has to assign scarce resources to complete the service request in time. Another component in real time database system is managing of the input streams of data and applying the corresponding update for tables in the database. In Industrial control systems input streams provides data from sensors, or service request in case of telecommunication system, call request or reflect the state of other databases in system. The main aim is to maintain external data consistency. If the size of materialize views is limited then the number of updates per second is so low that the resource demands for importing views are indeed trivial [2].

The scheduling problem in DSDM is very complex one it has considerable impact on the performance metrics

such as tuple latency and system throughput. Also the resources such as CPU, I/O and main memory are fixed but the scheduling jobs can be highly dynamic predefined quality of service requirements for query adds large constrains. The effective scheduling strategy in DSDM should be able to: (1) having fixed amount of resources, achieve the maximum performance. (2) If required guarantee the specified QoS for the query. (3)Take appropriate actions under conditions like unexpected overload. (4)Strategy should be implemented easily and run successfully when there is low overhead. The actual strategy may not able to fulfill all the above properties, because there are tradeoffs among all these performance metrics and usages of the available resources. The Path capacity strategy achieves best tuple latency. The segment scheduling strategy achieves the minimum memory requirement the simplified and threshold strategy provide reasonable overall performance though they do not meet all the properties [3].

To maximize the overall QoD (quality of data), when a set of materialized views are given the best order should be chosen to refresh the view. FIFO schedule for the web views updates may have disastrous effect of QoD. QoD-aware update scheduling algorithm (QoDA) that collaborate the scheduling of tables and view updates in one framework. QoDA scheduling can maintain a high level of QoD though the update processing capacity is not enough and there are surges in the incoming rate. QoD takes makes use of temporal locality in incoming update stream, also considers the database schema and support all types of views and arbitrary view hierarchies. In experimentation QoDA updates schedules continuously outperforms FIFO schedule by up to two orders of magnitudes, also FIFO never restores QoD or it may restore it slowly whereas QoDA rapidly restores the QoD [4].

To minimize the staleness of real time streaming warehouse the complexity of scheduling data-loading jobs is studied. Also the weighted staleness and stretch can be bounded under certain conditions on the process speed and the arrival time of new data. In the applications such as online financial trading, IP network monitoring and credit card fraud detection the streaming data warehouse gathers the large number of streams of data feeds that are generated by the external sources. For different tables data is generated at different rates, and for each table it is generated at constant rate. The arrival of each data triggers an n update and that appends the new data to the related table. When the processors are sufficiently fast the constant-stretch algorithm for the quasiperiodic model where tables can be clustered in few groups. Update frequencies within each group changes at most a constant factor. [5].

Earliest Deadline First (EDF) scheduling is the first dynamic priority-driven scheduling algorithms. It schedules real time jobs according to its priority. This priority can be assigned statically or dynamically by system. EDF suffers significantly when the system is overloaded. EDF algorithm is a dynamic and can be either online or offline, depending on the selection of the types of real-time jobs involved. Usually, offline scheduling has higher performance than online scheduling but may lead to poorer utilization. The metric of a real-time system is the success ratio of system deadlines. By success ratio the percentage of jobs completed before their deadlines. Other metrics, such as the minimized total Shortest Job First (SJF) scheduling is optimal but require expected execution time to fully implement. SJF can be implemented either non-preemptively or preemptively. SJF has low average waiting time. SJF is optimal with respect to average waiting time. This approach is good in comparison with other real-time algorithms [6].

2. STREAMING DATA WAREHOUSE

2.1 Streaming data warehouse architecture:

The streaming data warehouse architecture is shown in figure1. The warehouse maintains two types of table base and derived tables. The base table is loaded from the data stream and derived table is materialized view defined as an SQL Query on one or more than one tables. Each base or derived Table Tn has user defined priority Pn and time-dependent staleness function Sn(t) .The source and derived tables form a dependency graph of their relationship may be acyclic or directed. For each table Tn consider set of its ancestors that are directly and indirectly serve as its sources, the set of its dependent tables which are directly and indirectly sourced from Tn An update job Jn is released when new data arrive on stream n purpose of this job is to execute ETL process load new data into related table Tn, and update indices. When update for base table Tn is completed, update jobs are created for all tables directly sourced from Tn in order to disseminate new data that have been loaded into Tn. When those jobs are completed, update jobs for remaining tables are released in the BFS order specified in dependency graph. The purpose of the scheduler is to select the job which is going to be executing next. Each update job is modeled as nonpreemptible and atomic task, warehouse completes all the update jobs.

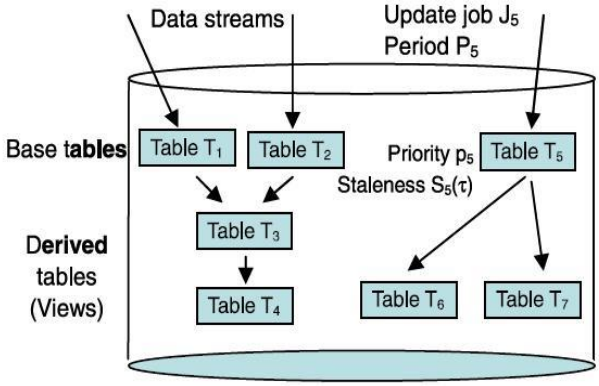

Fig 1 Streaming Dataware House

2.2 Data Staleness

Staleness of table Sn(t) at time t, is the difference between t and the freshness of Tn staleness begins to increase immediately after an update even if pending updates are not there. Priority assigned to table Tn is Pn goal is to minimize S that is the total priority-weighted staleness over time $S=\sum Pn \int Sn(t)dt$ Main objective is to minimize priority-weighted data staleness. [1]

2.3 Scheduling Model

Consider Ji be the new update job for table Ti, for base table Ji is equal to period of its source stream. For derived table Ji is equal to maximum period of any of Ti's ancestors. Fi freshness of Ti is equal to increase in freshness after Ji is complete. Consider n is time interval of data to be loaded

Execution time $Ei(n)=\alpha i + \beta i * n$
Where αi = Time to initialize the ETL process
βi = Data arrival rate

A new update job is released when batch of new data arrives, Hence multiple update jobs may be pending for same table, If warehouse is busy in executing another job. All such instances of pending update jobs are merged into single update job that loads all the available data into that table. It is more efficient than executing each such update job separately because we pay the fixed cost αi only once.

3. UPDATE SCHEDULING

An Update scheduling problem consists of finding the schedule of updates that maximizes the quality of data. Consider database with n relations having relations r1,r2,r3...rn and m views, v1, v2,v3, v4.....vm. Table1 has access frequencies fa() and cost to update each relation.

Object	Fa()	Cost	Type
r1	0	1	Relation
r2	0	1	Relation
v1	0.12	1	Mat. view
v2	0.37	2	Mat. view
v3	0.19	3	Mat. view
v4	0.09	1	Mat. view
v5	0.07	1	Mat. view
v6	0.06	1	Mat. view
v7	0.05	-	Virtual view

Table1- Frequency and Cost

Directed acyclic graph and view dependency graph can be used to represent derivative paths for views. The nodes of view dependency graphs lead to either relations or views. If node b is derived from node a then only there is edge between a and b. No other views are generated from virtual views. Consider for each relation we know the cost to update that relation and for each materialized view the cost to refresh that view the real update cost of is not needed but relative update cost are required. Figure2 shows dependency graph .for simplicity consider that all updates and refresh operations require one time unit except for views v2 and v3.at the end we have only two updates , one for r1 that arrives at time 0, and one for relation r2 that arrive at time 3.

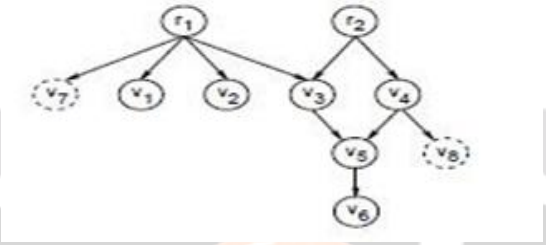

Fig 2 View dependency graph

In FIFO update propagation schedule, we should perform the refresh of all the affected views after the update to the parent relation is completed. When we have multiple paths for one view, we must avoid scheduling unnecessary refreshes.

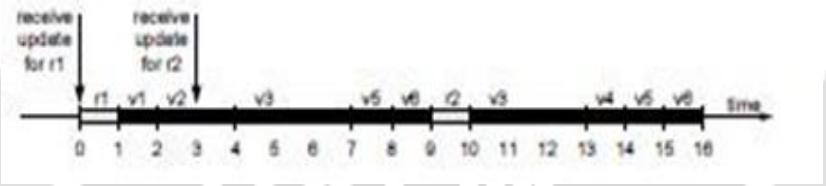

Fig 3 FIFO Update

For above figure 2 once we receive an update for r2, we would use schedule r2,v3,v4,v5 instead of r2,v3,v5,v4,v5, thus avoiding refreshing v5 twice. Using BFS technique on view dependency graph FIFO update schedule avoids unnecessary refreshes. Fig 3 FIFO update schedule for above example. The optimal update schedule has a 32.4% improvement over FIFO schedule, hence scheduling of updates has considerable impact on Quality of Data.

3.1 Scheduling Algorithm

Basic Algorithms: 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 look ahead can be chosen [1].Four scheduling algorithms are there for addressing the two closely related components: Do Update First(UF), Do Transaction 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 and 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 job set is heterogeneous with respect to execution times and periods, scheduler performance is likely to benefit if some part of the processing devices are guaranteed to small jobs. Global scheduling provides better results especially in a soft real time setting. Two methods are investigated for ensuring resources for small jobs: EDF-Partitioning, Proportional partitioning. EDF partitioned algorithm assigns jobs to tracks. The deadline of an update job is its release time plus period .EDF partitioning strategy is compatible with any local algorithm for scheduling on individual track. EDF partitioning strategy promotes short jobs to idle tracks that contain long jobs. The computational overhead is negligible in case of EDF partitioned algorithm. Proportional partitioning strategy identifies clusters of similar jobs .It can allocate job cluster to any number of tracks. The overhead of proportional algorithm is small. In worst case it varies according to availability of tracks.

3.3 Hierarchies of views

As views may depend on another view the prioritization of jobs becomes difficult. Source view need to be inherit the priority of their child view. Three ways of inheriting priority are Sum, Max, Max-plus.

3.4 Priority in the EDF Scheduling

EDF is implemented as dynamic priority driven scheduling algorithm. In real time system the number of priority levels should not exceed 32. EDF schedules the jobs based on deadlines. Hence one solution is to use one base priority and remaining dynamic priorities. The deadline of job finds its priority among all the jobs with the same level of base priority.

4. COMPARATIVE ANALYSIS

Scheduling the updates in streaming dataware house is discussed on large extend. Different metrics are used every time to show the best schedule. After the study of overall papers the idea of collaborating algorithm presented in paper 1 and 8 can enhance the performance of scheduling updates in streaming data warehouse. This idea can further lead to new area of research and implementation. Whereas the materialized views can be refreshed by choosing the best order to improve the quality of data. Number of basic algorithms is there to schedule the updates in streaming data warehouse. To scale the database global partitioning is used. There are two types EDF partitioning and proportional portioning. EDF algorithm with priorities is used for scheduling of the jobs.

5. 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 Propagation 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 Data depot," 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.