Open access Journal International Journal of Emerging Trends in Science and Technology

Ascendible Quantizing of Streaming Data Warehouse

Authors
Subba Reddy[1], Vikram Shinde[2]
[1]Assistant Professor, CSE Department, MLR Institute of Technology, Dundigal, Hyderabad, Telangana, India
Email: *subbareddy.lpu@gmail.com*
[2]CSE Department, MLR Institute of Technology, Dundigal, Hyderabad, Telangana, India
Email:*vickyshinde87@gmail.com*

Abstract
*In this paper we study update scheduling problem in streaming data warehouse, streaming data Warehouse combines the benefits of traditional data warehouses and data stream systems. In this problem jobs Corresponds to the processes which a load the new data in to the table, and main is aim is to minimize data staleness. Also it handle the challenges faced by streaming warehouse such as view hierarchies, preempt updates, data consistency and heterogeneity of update jobs caused by different arrival times and size of data. The scheduling metric is considered as staleness of data, we study the paper which explains how to schedule view updates in streams and transactions in soft real time database system, where two definitions of staleness are used one is MA (Maximum Age) and another is UU (unapplied Update). Four algorithms are examined for scheduling transactions And installing updates in soft real time database systems. Deadlines are very close To each other, SJF algorithm is used to schedule task within **a** group. Hence total execution time is minimized.*

Keywords: *about Ascendible quantizing, Data Stream management system, streaming data warehouse, QoD(Quality of Data),Strategy.*

Introduction

The data warehouse market is continuing to experience higher growth, primarily because of the role of data warehouse as a powerful decision support tool .Traditional data warehouses are updated during down times and store layers of complex materialized views over terabytes of historical data[1]. Whereas On the other hand, Data Stream Management Systems (DSMS) support simple analyzes on recently arrived data in real time. Data Depot, A streaming data warehouse, the goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible. When new data are loaded, the applications and triggers defined on the warehouse can take immediate action. For improving the efficiency of EDF algorithm in overloaded condition and in under loaded condition dynamic grouping of tasks (Jobs) is done whose deadlines are very close to each other and Shortest job first technique is used to sort the job within the group [2]. The business critical applications receives append only data streams from external sources. Recent work on streaming warehouse focus on high speed Extraction, Transformation & Load process. There has been also work on maintenance policies such as update views whenever the base changes & update view only when queried. However there has been some work on finding the out of date tables due to arrival of new data, and choosing that one for update next.

There has also been work on supporting various warehouse maintenance policies, such as immediate (update views whenever the base data change), deferred (update views only when queried), and periodic. Whenever the update for base table arrives, we update the corresponding base table T immediately. After updating of base table, we perform the updates of all the materialized views whose source table is T, also all the views followed by defined over those views, and so on. The problem is new data may arrive on multiple streams, but there is no procedure for limiting the number of tables that can be updated simultaneously. Running no of parallel updates can degrade performance due to memory and CPU-cache thrashing (multiple memories intensive ETL Processes are likely to exhaust virtual memory), disk-arm overwhelm etc. system model, formalize the notion of staleness, and describe the solution for maintaining data consistency under arbitrary update schedules. In this scheme a multitrack algorithm orders jobs by some reasonable means. This algorithm can be expanded to handle the complications encountered by a streaming warehouse. In case of to scale the large and diverse job sets, this system combines the efficiency of global scheduling with the guarantees of partitioned scheduling In partitioned scheduling, we group the update jobs by their execution times. Each group defines a partition and runs its own local algorithm. Normally, at most one job from each partition can run at any given time. on the other hand under certain conditions, we allow some pending jobs to be scheduled on a different tracks if their “home” track is busy. Thus, we can reserve processing resources for short jobs while achieving varying degrees of global scheduling.

Applications include:

1. Online stock trading in which new transactions generated by various stock exchanges and compared against historical trends in nearly real time to gain maximum profit.
2. Monitoring applications which use sensor networks in which complex filtering and activation of alarms required for unusual conditions.
3. Network traffic analyses to monitor network performance and detect network attacks all is done by network data warehouses that are maintained by Internet service providers. Also these ISPs collect various system logs.
4. Financial tickers that is online analysis of stock prices that involves co-relations , identifying trends and arbitrage opportunities also forecasting future values. 5 Transactional log analysis: Online mining of web usage logs, automated bank machine transactions and telephone call records.

Contributions and work

We consider the update scheduling problem in streaming warehouses. In topic 2 we discussed related work to this as literature survey, In topic 3 we introduce system model which consists of streaming data warehouse, system architecture. Our topic 4 presents our multitrack scheduling algorithm and the two strategies which are merged in our algorithm. Topic 5 presents experimental results. We find that our multitrack algorithm schedules jobs on track efficiently and hence the track utilization is better than tradition algorithm and topic 6 conclude the paper.

Streaming data warehouse update problem is modelled as scheduling problem in which jobs corresponds to processes. These processes load new data into tables, and aim is to improve data freshness. The scheduling decisions depends on effect of update job on data staleness. Average staleness was considered as scheduling metric and the scheduling algorithm were designed to handle the environment of streaming data warehouse[1]. The proposed system is algorithm which uses a two-level scheduling strategy for scheduling non-pre-emptive tasks. The dynamic grouping of tasks is done and shortest job first (SJF) algorithm is used for scheduling the jobs within the groups[2] To keep the database fresh a real time database system must process new updates in timely manner, also at the same time transaction should processed with their time constraints. Four algorithms for scheduling of transactions and processing of updates were designed[3].Data warehouse can be divided into two parts for ensuring consistency: ensuring that each view reflects a consistent stare of base table, ensuring that multiple views are mutually consistent. Guarantying multiple view consistency, Identify and define three layers of consistency for materialized views in distributed systems.[4] A method to refresh to a local copy of autonomous data source to maintain the copy up to date. As the data size goes increasing ,it becomes difficult to maintain the fresh copy making it crucial to synchronize the copy electively. Change model of the underlying data and synchronization policies as the two fresh metrics used.[5].An operator scheduling strategy, chain scheduling for data stream systems that is nearly optimal in runtime memory usage for any collection single stream queries involving projections, selections, foreign-key joins with stored relations. Chain scheduling can also be used for queries with sliding window joins over multiple streams.[6].When a database and the set of relations that are materialized also the incoming relation update stream is provided , the update schedule that maximizes the overall quality of data. Lets consider a database with two relations r1, r2 & eight materialized views v1.....v8. Views v1 through v6 are materialized. Also assume all updates and refresh operations take 1 time unit for views excluding view v2 and v3.Here we have only two updates one for relation r1 at time 0 and for r2 that arrive at time 3. Using FIFO update propagation schedule all the affected views can be refreshed. When update for base relation is completed. The FIFO update schedule avoids unnecessary refreshes .FIFO performs breath first search traversal of the view dependency graph to compute the refresh order [7].

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 may be partially or fully. The base table is directly loaded from the data stream whereas the derived table is materialized view defined on one or baser or derived tables. The base and derived table Tj has user defined priority Pj also time dependent function Sj(t).Relationship between base table and derived table can be defined by dependency graph the may be acyclic and directed.[1] Each data stream i is generated by external sources, with a batch of new data , consisting of one or more records that are being pushed to the warehouse with period Pi. If the period of a stream is unknown , then the user can choose the period with which the new data should checked by warehouse When new data arrive on stream i an update job Ji is release means Ji is inserted into the scheduler queue. Job executes ETL process, loads new data in to the corresponding base table Ti, and update any indices. When update for base table is done number of update jobs are released for update in derived tables of that source table Ti. The aim of the update scheduler is to decide which of the released update job to execute next. If multiple update for a given table are pending, then they must be completed in sequential manner as their arrival time.

B. System Architecture

This paper presents a model as shown in the fig 1 When the updates for the table arrives on stream , it is converted into jobs. These jobs are then sorted according to their arrival time using PEDF algorithm. The sorted list of jobs is then divided into suitable clusters depending on some criteria.(such as jobs having arrival time in between 1 to 5 goes in first cluster and so on). All the jobs within one cluster are sorted according to SJF (Shortest job first) algorithm of their execution time and average data.

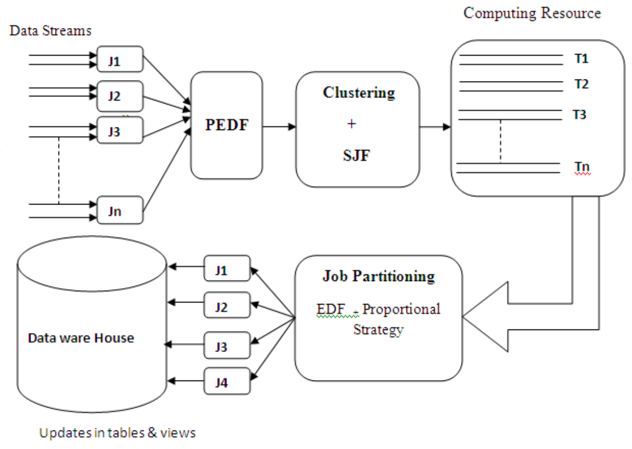


Fig 1: System Architecture

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

By combining some features of these two strategies our multitrack algorithm proves that the performance is better than using only EDF partitioning Strategy. When all the jobs are scheduled & executed by multitrack algorithm the data warehouse updates are done simultaneously.

Scheduling Model

For table Ti the update job is Ji, consider for base table the period of Ji and the period of its source stream are same. The freshness of base table can be defined as increase in freshness after completion of Ji. Let n is equal to time interval of data to be loaded , execution time of update job Ji =αi+ßi*n Where αi = Time for initialization of ETL process and ßi = Rate at which data arrives. If no of updates are there for same table then all the updates are merged into single job that loads all the available data into that table.

A. Multitrack Scheduling Algorithm:

The update jobs are partitioned according to their expected processing time. The computing resources are partitioned into tracks.. When any update job is release it is placed into queue of respective partition. Here scheduling decisions are made by scheduler. The jobs are priorities by their local algorithm on individual tracks. Multitrack scheduling algorithm is as follows:

1. Create the jobs of updates which appears in stream format.
2. Sort the jobs using PEDF algorithm.
3. Create no of clusters and add jobs to that clusters respectively.
4. Sort the jobs within the cluster by SJF algorithm and assign priorities to jobs.
5. Divide the computing resource in logical tracks.
6. Partition the jobs by using EDF and proportional partitioning strategy.
7. Update the data ware house.

We need some additional terminology for this multitrack algorithm:

- Emax= Execution time of longest job, Maximum processing time of job. Within track
- Pmin= Minimum period of the job within the track.
- U= Track utilization.
- Total track utilization =Σ iUi

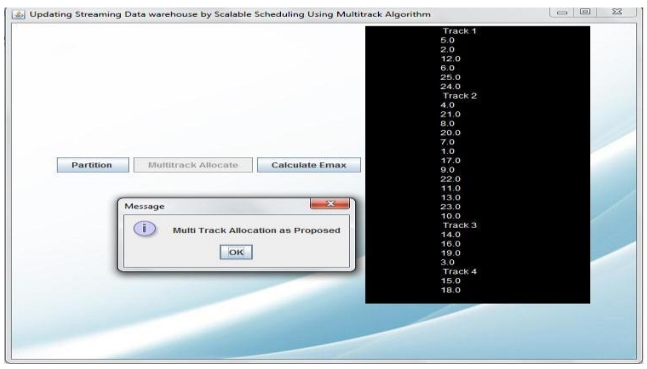
Using this multitrack algorithm we make the updates in the data warehouse more faster than the other methods. We combine This EDF and Proportional Partitioning strategy together in multitrack algorithm.

EDF Partitioning algorithm is as follows: 1 Sort the released jobs by the local algorithm. 2. For each job Ji in sorted ordera. If Ji's home track is available, schedule Ji on its home track.

b. Else, if there is an available free track, schedule Ji on the free track. c. Else, scan through the tracks r such that Ji can be promoted to track r i. If track r is free and there is no released job remaining in the sorted list for home track r, A. Schedule Ji on track r. 3. Else, delay the execution of Ji. **Proportional Partitioning strategy step1:** 1. Order the jobs by 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. Add Ji to the current cluster. b. Else, create a new cluster, make it the current cluster, and add Ji to it. 4.For each cluster Cj A compute UCj=Σ Ei(Pi)/Pi, ranging over tasks in Cj 5.Compute total utilization= ΣCj.

Experimental Results

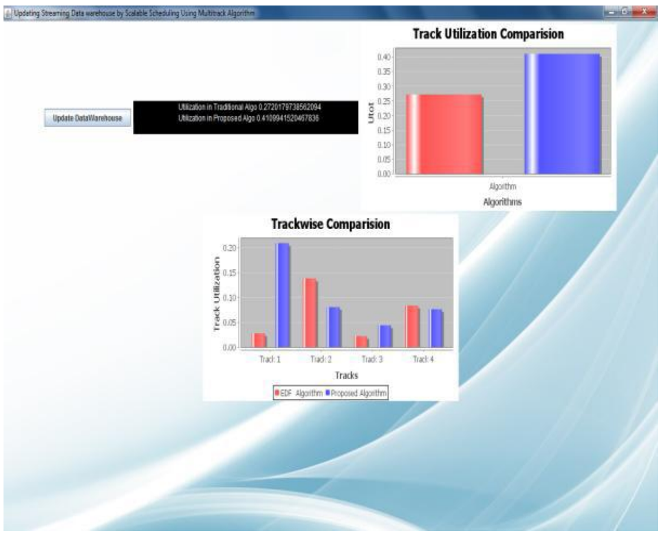
The updates received are converted into jobs, that jobs are then sorted according to their arrival time, by PEDF algorithm. The sorted list is then clustered into four groups, the jobs which satisfy the criteria of the cluster is added in that cluster, After that the shortest job first algorithm is used to sort the jobs within the cluster. Hence the jobs having minimum average data are placed at the top position. Then the multitrack algorithm uses the combined partitioning strategies (EDF& Proportional) and jobs are assigned to different tracks accordingly.(Here we have taken 4 tracks). Now we calculate the values for Emax and Pmin and we calculate the track utilization using multitrack algorithm and also track utilization using EDF partitioning algorithm finally on comparison it is seen that multitrack algorithm gives better results than EDF partitioning algorithm. The below screen shows result of the multitrack algorithm. The jobs are scheduled on multiple tracks.


Updating Streaming Data warehouse by Scalable Scheduling Using Multitrack Algorithm
Partition
Multitrack Allocate
Calculate Emax
Message
OK

Multi Track Allocation as Proposed

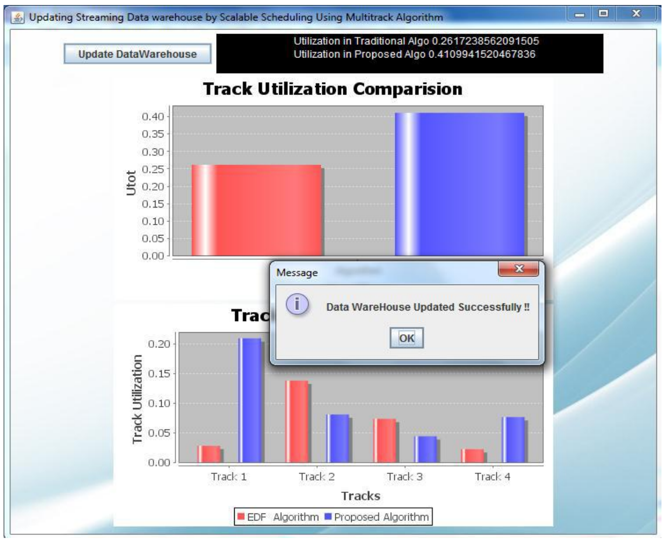
```
Track 1
5.0
2.0
12.0
6.0
25.0
24.0
Track 2
4.0
21.0
8.0
20.0
7.0
1.0
17.0
9.0
22.0
11.0
13.0
23.0
10.0
Track 3
14.0
16.0
19.0
3.0
Track 4
15.0
18.0
```

The below screen shows comparative result of the multitrack algorithm and EDF portioning strategy on samesorted list of jobs. The multitrack algotithm performs better than EDF algorithm. As the track utilization is more the updates are done faster. Hence we can update streaming data warehouse tables faster if we use multitrack algorithm to schedule the jobs. In below snap shot it is shown that the track utilization of EDF is 0.2147 and track utilization of multitrack algorithm is 0.41099. Also it shows the update data warehouse button and track wise comparison of EDF and multitrack algorithm where EDF is represented as RED colored bar and Multitrack is represented as BLUE bar.



Utilization in Traditional Algo 0.2720179738562094
Utilization in Proposed Algo 0.4109941520467836

The below snapshot shows that when we click on Update Data Warehouse all the jobs are exexuted according to their secheduled sequence and finally it shows the message “Data WareHouse Updated Successfully!!”



Utilization in Traditional Algo 0.2617238562091505
Utilization in Proposed Algo 0.4109941520467836

Data WareHouse Updated Successfully !!

Conclusion

In this paper we solved the problem of non preemptively scheduling updates in streaming data warehouse. We propose the multitrack algorithm that convert the update queries into jobs, sorts the jobs according to their arrival time and execution time and finally schedule the jobs on multiple tracks so that the jobs are executed as early as possible hence short jobs are not suffered from long jobs. As the track utilization is higher in case of multitrack algorithm the updates are done proportionally. As future work we plan to improve the efficiency of our multitrack algorithm.

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. 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
3. 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.
4. Qingchun Jiang, Sharma Chakravarthy," Scheduling Strategies for a Data Stream Management System".
5. 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.
6. 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.

Author's Profile

Subba Reddy, B.Tech, M.Tech CSE. He is currently working in the Department of Computer Science and Engineering, MLRIT, Telangana, India. His research interesting areas Programming (C&DS, C++, and JAVA), Data Mining, Software Engineering, Network Security & Computer Networks.

Vikram Shinde Currently Pursuing Master of Technology in CSE from MLR Institute of Technology, Dundigal, Hyderabad, Telangana, India