International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 02 | Feb-2016 www.irjet.net p-ISSN: 2395-0072

Scalable scheduling of updates in streaming data warehouses

Mr. Nikhil Sangar[1] , Mr. Abhijit Nikam[2], Mr. Vikas Lokhande[3], Mr. Pradip Chougule[4]

[1] *Dept. of Computer Engineering, Dr. J J Magdum College Of Engineering, jaysingpur*
[2] *Dept. of Computer Engineering, Dr. J J Magdum College Of Engineering, jaysingpur*
[3] *Dept. of Computer Engineering, Dr. J J Magdum College Of Engineering, jaysingpur*
[4] *Assistant Professor, Dept. of Computer Engineering, Dr. J J Magdum College Of Engineering, jaysingpur*

Abstract - *We discuss update scheduling in streaming data warehouses, which combine the features of traditional data warehouses and data stream systems. In our setting, external sources push append-only data streams into the warehouse with a wide range of Inter arrival times. While traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. We model the streaming warehouse update problem as a scheduling problem, where jobs correspond to processes that load new data into tables, and whose objective is to minimize data staleness over time (at time t, if a table has been updated with information up to some earlier time r, its staleness is t minus r). We then propose a scheduling framework that handles the complications encountered by a stream warehouse: view hierarchies and priorities, data consistency, inability to preempt updates, heterogeneity of update jobs caused by different inter arrival times and data volumes among different sources, and transient overload. A novel feature of our framework is that scheduling decisions do not depend on properties of update jobs (such as deadlines), but rather on the effect of update jobs on data staleness. Finally, we present a suite of update scheduling algorithms and extensive simulation experiments to map out factors which affect their performance.*

Key Words: (Size 10 & Bold) Key word1, Key word2, Key word3, etc (Minimum 5 to 8 key words)…

1.INTRODUCTION

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.

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.

1.1 Scheduling

The closest work to ours is, which finds 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 from Labrinidis and Roussopoulos, 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. Another closely related work is which studies the complexity of minimizing the staleness of a set of base tables in a streaming warehouse. In general, interesting scheduling problems are often NP hard in the offline setting and hard to approximate offline. This motivates the use of heuristics such as our greedy Max Benefit algorithm. While we believe that update scheduling in a streaming warehouse is novel, our solution draws from a number of recent scheduling results. In particular, there has been work on real-time scheduling of heterogeneous tasks on a multiprocessor to address the tension between partitioned scheduling and global scheduling. The Pair algorithm and its variants have been proposed when tasks are perceptible, however we must assume that data loading tasks are non pre-emptable. Our Proportional algorithm attempts to make a fair allocation of resources to non pre-emptible tasks in a multi track setting, and is the first such algorithm of which we are aware.

1.2 Data Warehousing

There has been some recent work on streaming data warehousing, including system design, real-time ETL Processing, and continuously inserting a streaming data feed at bulk-load speed. These efforts are complementary to our work they also aim to minimize data staleness, but they do so by reducing the running time of update jobs once the jobs are scheduled. A great deal of existing data warehousing

research has focused on efficient maintenance of various classes of materialized views, and is orthogonal to this paper. In and discuss consistency issues under various view maintenance policies. As discussed earlier, maintaining consistency in a streaming data warehouse is simpler due to the append-only nature of data streams. There has also been work on scheduling when to pull data into a warehouse to satisfy data freshness guarantees. This work does not apply to the push based stream warehouses studied in this paper, which do not have control over the data arrival patterns. Quantifying the freshness of a data warehouse was addressed in several works.

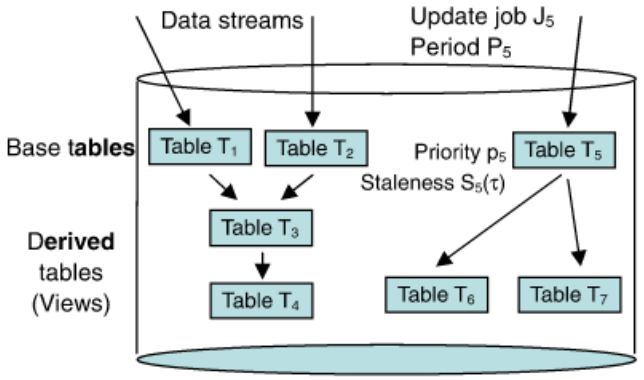

Fig. 1. A streaming data warehouse.

Table -1: Symbols Used In This Paper

Symbols	Meaning
Ps	Period of stream s
Ji	Update job for table i
Ei(n)	Execution time of Ji on data produced in a time interval of length n
Ri	Release time of Ji
Pi	Priority of table i
Fi(r)	Freshness of table i at time r
Si(r)	Staleness of table I at time r
△F1	Freshness delta of table i
Ai	Time to initialize the ETL process for table i
Bi	Data arrival rate to table i

Data Stream Management

One important difference between a DSMS and a data stream warehouse is that the former only has a limited working memory and does not store any part of the stream permanently. Another difference is that a DSMS may drop a fraction of the incoming elements during overload, whereas a streaming data warehouse may defer some update jobs, but must eventually execute them.

2. SCHEDULING ALGORITHMS

2.1. Basic Algorithms

The basic scheduling algorithms prioritize jobs to be executed on individual tracks, and will serve as building blocks of our multi track solutions. For example, the Earliest-Deadline-First (EDF) algorithm orders released jobs by proximity to their deadlines. EDF is known to be an optimal hard real-time scheduling algorithm for a single track (w.r.t. maximizing the number of jobs that meet their deadlines), if the jobs are pre emptible. Since our jobs are prioritized, using EDF directly does not result in the best performance. Instead we use one of the following basic algorithms.

Prioritized EDF (EDF-P) orders jobs by their priorities, breaking ties by deadlines. Our model does not directly have deadlines, but they may be estimated as follows: For each job ,we define its release time as the last time 's freshness delta changed from zero to nonzero (i.e., the last arrival of new data in case of base tables, or, for derived tables, the last movement of the trailing edge point of its source tables).

2.2. Job Partitioning

If a job set is heterogeneous with respect to the periods and execution times (long execution times versus short periods), scheduler performance is likely to benefit if some fraction of the processing resources are guaranteed to short jobs (corresponding to tables that are updated often, which generally have higher priority). The traditional method for ensuring resource allocation is to partition the job set and to schedule each partition separately [7] (and to repartition the set whenever new tables or sources are added or existing ones removed, or whenever the parameters of existing jobs change significantly). However, recent results indicate that global scheduling (i.e., using a single track to schedule one or more jobs at a time) provides better performance, especially in a soft real-time setting, where job lateness needs to be minimized. In this section, we investigate two methods for ensuring resources for short jobs while still providing a degree of global scheduling:
EDF-Partitioned and Proportional.

EDF-Partitioned Strategy

The EDF-partitioned algorithm assigns jobs to tracks in a way that ensures that each track has a feasible non-preemptive EDF schedule. A feasible schedule means that if the local scheduler were to use the EDF algorithm to decide which job to schedule next, all jobs would meet their deadlines. In our setting, we assume that the deadline of an update job is its release time plus its period, i.e., for each table, we want to load every batch of new data before the next batch arrives.

Proportional Partitioning Strategy

The EDF-partitioned algorithm has some weaknesses. For one, a collection of jobs with identical periods (and perhaps

Identical execution times) might be partitioned among several tracks. The track promotion condition among these jobs and tracks is the same as the condition which limits the initial track packing—and therefore no track promotion will be done. We can patch the EDF-partitioned algorithm by using multi track schedulability conditions, but instead we move directly to a more flexible algorithm.

3. CONCLUSIONS

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. We have implemented some of the proposed algorithmsin the Data Depot streaming warehouse, which is currently used for several very large warehousing projects within AT&T. As future work, we plan to extend our framework with new basic algorithms. We also plan to fine-tune the Proportional algorithm—in our experiments, even the aggressive version with "all" allocation still exhibits signs of multiple operating domains, and therefore can likely be improved upon (however, it is the first algorithm of its class that we are aware of). Another interesting problem for future work involves choosing the right scheduling "gran-ularity" when it is more efficient to update multiple tables together, as mentioned. We intend to explore the tradeoffs between update efficiency and minimizing staleness in this context.

ACKNOWLEDGEMENT

The authors would like to thank Our Guide & HOD Madam for their helpful discussions, and they thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] 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.

[2] 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.

[3] S. Babu, U. Srivastava, and J. Widom, "Exploiting K-constraints to Reduce Memory Overhead in Continuous Queries over Data Streams," ACM Trans. Database Systems, vol. 29, no. 3, pp. 545- 580, 2004.

[4] S. Baruah, "The Non-pre-emptive Scheduling of Periodic Tasks upon Multiprocessors," Real Time Systems, vol. 32, nos. 1/2, pp. 9- 20, 2006.

[5] S. Baruah, N. Cohen, C. Plaxton, and D. Varvel, "Proportionate Progress: A Notion of Fairness in Resource Allocation," Algorithmica, vol. 15, pp. 600-625, 1996.