Predictive Memory Allocation over Skewed Streams

Hong-Won Yun, *Member, KIMICS*

Abstract—Adaptive memory management is a serious issue in data stream management. Data stream differ from the traditional stored relational model in several aspect such as the stream arrives online, high volume in size, skewed data distributions. Data skew is a common property of massive data streams. We propose the predicted allocation strategy, which uses predictive processing to cope with time varying data skew. This processing includes memory usage estimation and indexing with timestamp. Our experimental study shows that the predictive strategy reduces both required memory space and latency time for skewed data over varying time.

Index Terms—Data Stream Management System, Adaptive Memory Management, Skewed Data Streams, Continuous Queries.

I. INTRODUCTION

There are many stream-processing applications, including financial record analysis, traffic data monitoring, network monitoring, and sensor data monitoring. For supporting continuous queries on massive data streams with real time or almost real time response, memory efficiency can be a serious issue due to the large volume of data streams [1-3].

For applications monitoring and processing streams, we consider in particular continuous queries, which are queries that are issued once and the logically run continuously over the data streams. Continuous queries may be used to monitor network behavior and trends in financial applications. The potential for high data arrival rates and large data volumes make memory and execution time performance of stream query evaluation critical. Stream aggregation using windowed aggregate queries is a common operation on data streams [4].

In general, data streams are usually massive and arrive with high speed, making either storing all the historical data scanning it nearly not easy. Moreover, stream data often evolve considerably over time. In many applications, massive data streams often have data distributions that result in a few dense aggregate groups and a tail of many sparse aggregate groups [3,5,6].

In this paper we focus on improving memory efficiency with skewed data distributions, when they vary over time. The paper is organized as follows. We first introduce the background of data stream features and stream semantics in Section II. Then in Section III we present our strategy for efficient memory allocation. Next we show performance evaluation of our proposed strategy in Section IV. Finally we conclude our study and discuss the related issues in Section V.

Manuscript received May 20, 2009 ; Revised June 8, 2009. Hong-Won Yun is with the Department of IT, Silla University, Busan, 617-736, Korea (Tel: +82-51-999-5065, Fax: +82-51-5657, Email: hwyun@silla.ac.kr)

II. RELATED WORK

A. Stream Properties

Traditional DBMS's are not designed for rapid and continuous loading of individual data items, and they do not directly support the continuous queries that are typical of data stream applications. Some of all of the input data streams that are to be operated on are not available for random access from disk or memory, but rather arrive as one or more continuous data streams. Data streams differ from the traditional stored relation model in several ways [7]:

- The data streams arrive online continuously.
- The system can not control over the order which data streams arrive to be processed.
- Data streams size is not bounded.

Operating in the data stream model does not preclude the presence of some data in conventional stored relations. Queries over continuous data streams have much in common with queries in a traditional DBMS. There are significant distinctions peculiar to the data stream model. Continuous queries are evaluated continuously as data streams continue to arrive. For example, aggregation is the process of

computing statistical measure such as means and variance that summarize the incoming stream. The problem with aggregation is that it does not perform well with highly fluctuating data distributions [7,8].

B. Streams Semantics

Data streams are generated at distributed sources, continuous queries registered with the DSMS are evaluated over the incoming stream data. In most of stream applications, each data volume is varied over time and massive data streams often have skewed data distributions. Queries over data streams have much in common, for example, stream aggregation using windowed aggregate queries is a common operation on data streams. The semantics for continuous queries in a data stream system typically assumes timestamps on data stream elements. Stream tuples are timestamped on entry the DSMS using DSMS system time [9-12].

Since data streams are potentially unbounded in size, the amount of storage required to compute an exact answer to a data stream query may also grow without bound. Policies for memory efficiency can be a significant issue due to the large volume of some data streams. We formalize the problem and define the probabilistic prediction method for memory efficiency. We present a timestamp B-tree index with weight of each data sources that process continuous queries over data streams.

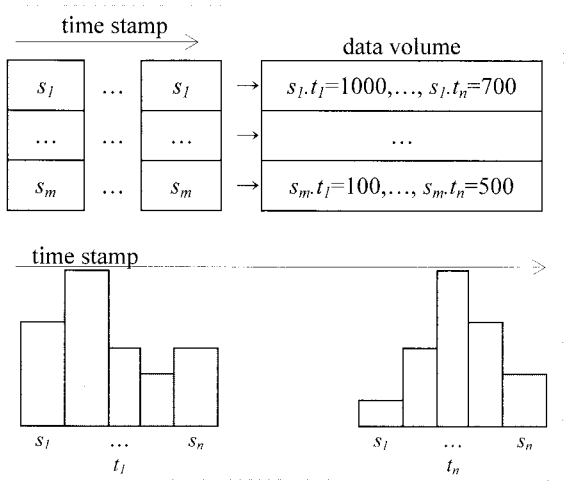


Fig. 1 Skewed data distributions in data streams

III. MEMORY USAGE PREDICTION AND INDEXING

A. Memory Usage Estimation

Memory and execution performance of stream query evaluation is critical in the potential for high data arrival rates and huge data volumes with skewed distributions. We need to estimate the memory usage of each data sources over varying time.

Stream volume describes the mount of data in a stream. We define data volume, V, as amount of data per specific time interval, T, in total windowing attribute. The unit of time can be from second to year denoted as t and it is changed $t_1, \ldots, t_m$. Let $D = \{d_1, \ldots, d_m\}$ be an amount of data that appear in the total target data sources. The amount of data in the specific time interval is denoted as following:

$$V(D,T) = \sum_{di \in D} V(di,T) = \sum_{tj \in T} V(D,tj) = \sum_{di \in D} \sum_{tj \in T} V(di,tj) \quad (1)$$

Each data source has an amount of data v in a time interval, T. It is denoted as $v(d,T)$. Let $d = \{e_1, \ldots, e_k\}$ be a set of data in the each data source. Thus,

$$v(d,T) = \sum_{ek \in d} v(ek,T) \quad (2)$$

The amount of data in all data sources is

$$V(D,T) = \sum_{vk \in V} vk(d,T) \quad (3)$$

We use $V(d_i)$ for $V(d_i,T)$ and $V(D)$ for $V(D,T)$. An amount of random data source, $W(v_i)$, within total volume of data V is estimated as:

$$W(vi) = V(di)/V(D) \quad (4)$$

In the Fig. 2, $t_1, \ldots, t_m$ present unit of time in the time interval T and $s_1, \ldots, s_m$ mean data sources in the observed system. Summation of the total amount of data sources is divided by the total volume of data will be 1 approximately.

Time / Sources	t_1	...	t_m
s_1	$W(v_1)$	...	...
...	...	...	...
s_m	...	...	$W(v_m)$

Fig. 2 Amount of each data sources within total data volume. Each amount is represented weight

B. Indexing

By using probabilistic, the system copes with the skewed data volumes over varying time. The distribution is specified separately for the time attribute and the data sources. The processor can use

indices to allocate the memory for each data sources that are skewed on changing time. Prediction stream indexing must properly handle probabilistic value using historical data.

Fig. 2 shows one possible index for efficiently identifying skewed data sources that are relevant to the time. This indexing approach on (data source, time) allows efficient look up of time steps in which the node has specific probability.

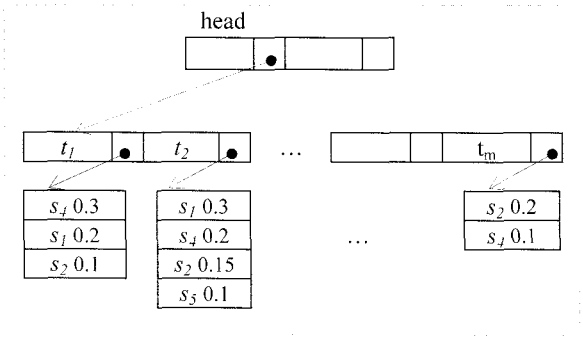


Fig. 3 Timestamp B-tree index with weight of each data source

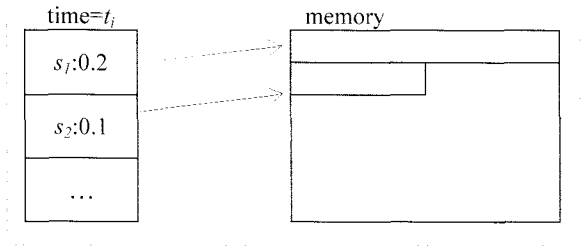


Fig. 4 Predicted memory allocation at time t

IV. PERFORMANCE STUDY

A. Experimental Environment

We tested the effectiveness and efficiency of predicted memory allocation using probabilistic method by conducting experiments. Our experiments were conducted on an Intel Pentium 4 3GHz machine, running Windows XP, with 1 GB main memory.

Fig. 5 illustrates the simplified schematic showing the simulation setup for our experiments. To vary the data skew, we distributed 1 percent, 3 percent, 5 percent, 7 percent, or 9 percent of the data. The data set size is roughly 155 Megabytes. We used five data sources to simulate, each data source generated streams that had delta between 1 percent and 9 percent. Data source s1 has small skewed data distribution and then data source s5 has large skewed data distribution. Each data source is assumed to have a unique source identifier, and every incoming tuple has timestamp.

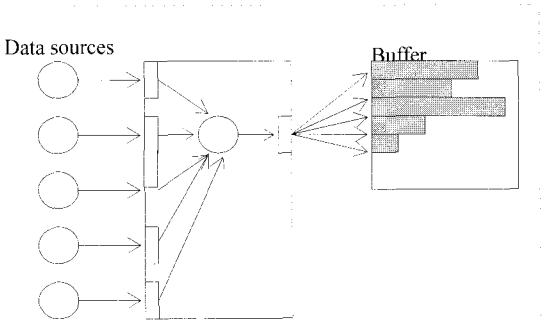


Fig. 5 Schematic showing the simulation setup

Table 1 Skewed Data Distribution (%)

	s1	s2	s3	s4	s5
Δ=1	1	2	3	4	5
Δ=3	3	6	9	12	15
Δ=5	5	10	15	20	25
Δ=7	7	14	21	28	35
Δ=9	9	18	27	36	45

B. Results

We present the results of the two different experiments. The experiments used rates and varied the parameters according to Table 1. Fig. 6 and 7 show the results of our experiments using the aggregate function. In Fig. 6, the memory usage of Origin a little increases and has maximizing memory usage as we change the difference of skewed data from data source 1 to source 5. We see from Fig. 6 that the amount of memory used by the Prediction method is greatly reduced. The Prediction method can significantly reduce memory usage. This is because the memory usage prediction can be more adaptive for skewed data streams over varying time than the original method for memory allocation.

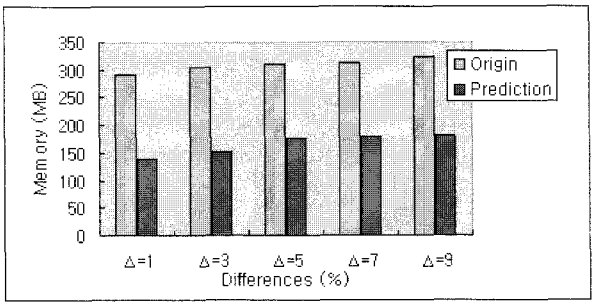


Fig. 6 Memory usage over data-skewed sources

As Fig. 7 shows, the Prediction outperforms the Origin in all five differences that mean the percentage of skewed data: the Prediction's latency performance is significant, confirming our expectations. Its latency

benefit increase mainly because the Prediction reduces the waiting time to allocate memory for each data sources on varying time. The incoming data streams are largely skewed with individual sources, the latency time has steeper increasing. The Prediction strategy show stable latency performance with increasing difference, but the Origin is affected more severely. This improvement with the Prediction is because forcing more adaptive memory allocation when the skewed data grows over varying time.

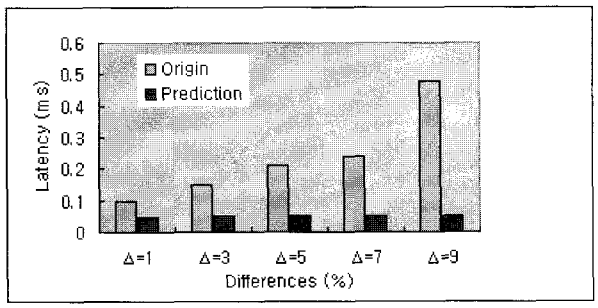


Fig. 7 Latency performance over data-skewed sources

IV. CONCLUSIONS

Adaptive memory management is of particular importance for a data stream management system since continuous queries over data streams as well as changing stream properties over varying time. Memory efficiency can be an important issue due to the huge volume of data streams. Data skew is a common feature of massive data streams. In this paper we have formalized the problem and defined the probabilistic prediction method for memory efficiency.

Also we have presented a timestamp B-tree index with weight of each data sources that process continuous queries over data streams. This indexing approach on data source and time allows efficient look up of time steps in which the node has specific probability. Our proposed strategy can optimize for memory performance and latency performance over data skewed sources. In general, we believe this strategy will be efficient in any application in which stream characteristics can be skewed statistically over varying time within a specific time interval.

REFERENCES

[1] P. A. Tucker, D. Maier, T. Sheard, and L. Fegaras, "Exploiting Punctuation Semantics in Continuous Data Streams," *IEEE Trans. On Knowledge and Data Engineering*, Vol. 15, No. 3, pp. 555–568, 2003.

[2] S. Badu, U. Srivastava, and J. Widom, "Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams," *ACM Trans. On Database Systems*, Vol. 29, No. 3, pp. 545–580, 2004.

[3] J. Li, K. Tufte, D. Maier, and V. Papadimos, "AdaptWID: An Adaptive, Memory-Efficient Window Aggregation Implementation," *IEEE Internet Computing*, Vol. 12, No. 6, pp. 22–29. 2008.

[4] M. Cammert, J. Kramer, B. Seeger, and S. Vaupel, "An Approach to Adaptive Memory Management in Data Stream Systems," *Proc. of ICDE '06*, pp. 137–139, 2006.

[5] L. Golab and M. T. Ozsu, "Issues in Data Stream Management," *SIGMOD Record*, Vol. 32, No. 2, pp. 5–14, 2003.

[6] F. Wang and P. Liu, "Temporal Management of RFID data," *Proceeding of the VLDB '05*, pp.1128–1139, 2005.

[7] B. Babcock et al., "Model and Issues in Data Stream Systems," *Proc. Symp. Principles of Database Systems*, ACM Press, pp. 1–16, 2002.

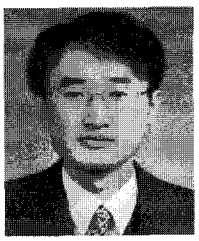
[8] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining Data Streams: A Review," *ACM SIGMOD Record*, Vol. 34, No. 2, pp. 18–26, 2005.

[9] U. Srivastava and J. Widom, "Flexible Time Management in Data Stream Systems," *PODS 2004*, ACM, pp. 263–274, 2004.

[10] J. Gao et al., "Classifying Data Streams with Skewed Class Distributions and Concept Drifs," *IEEE Internet Computing*, Vol. 12, No. 6, pp. 37–49. 2008.

[11] J. Li et al., "Semantics and Evaluation Techniques for Window Aggregates in Data Streams," *Proc. ACM SIGMOD 05*, ACM Press, pp. 311–322, 2008.

[12] D. Abadi et al., "Aurora: A New Model and Architecture for Data Stream Management," *VLDB J.*, Vol. 12, No. 2, pp. 120–139, 2003.



Hong-won Yun

He received his B.S. and the Ph.D. degrees at the Department of Computer Science from Pusan National University, Korea, in 1986 and 1998, respectively. He is a professor at the Department of Information Technology, Silla University in Korea. His research interests include temporal database, semantic web, and data stream management system.