DATA STREAMS MANAGEMENT IN THE REAL-TIME DATA WAREHOUSE: FUNCTIONING OF THE DATA STREAMS PROCESSOR

M. Shoaib, F. Majeed, Shazia, K. Kalsoom, S. Majid** and S. Mahmood*

Dept. of CS & E, University of Engineering & Technology, Lahore, *Dept. of CS, CIIT Lahore, Pakistan
**Department of Computer Science , LCWU, Pakistan
Corresponding author email: shoaib_uet@hotmail.com

ABSTRACT: Data stream applications are currently emerged as sources of Real Time Data Warehouse (RTDW) systems. Existing Extraction, Transformation and Loading (ETL) tools have not been crafted for fast, continuous and time-varying data streams. To address this, we have already proposed a framework of RTDW to efficiently capture process and load the data streams. This article provides technical architecture of Data Streams Processor, the main component of the framework. It discusses approximations in detail including data structures and processes to manage fast data streams in memory. Moreover, algorithm for approximations is presented and clarified with an example. Finally, Data Streams Processor is applied on a real-world case study. (This paper has been extracted from the thesis of Fiaz Majeed, 2009)

Key words: Real-time Data Warehouse; data stream; strategic decision-making; tactical decision-making; Data Streams Processor

INTRODUCTION

A lot of literature has been published on the Data Warehouse. The traditional Data Warehouse stores consolidated, time variant, read-only and integrated data for analysis of decision-makers (Inmon, 1996; Ponniah, 2001). The data is extracted from source systems in batches (daily or weekly). It is then transformed (cleansing and merging etc.) to renovate data in the Data Warehouse schema format. Finally, data is loaded in the Data warehouse from load files using loading utility. These mentioned processes are performed by ETL tools (Mohanty, 2006). Several diverse processes are involved that work co-operatively to make functioning the Data Warehouse. Such processes include integration of extract files, cleansing, management and controlling module, metadata repository, summary table generation, index and view materialization, front-end tools and so on (Ponniah, 2001).

With the demand of tactical decision-making, active or real-time data warehouse was emerged (Basu, 2003; Broast, 2002). The data from operational sources is loaded simultaneously when new data made available (Raden, 2003). With this innovation, tactical decision-making was provided as well as strategic decision-making (Vandermay, 2000). Time is important element for the Data Warehouse records. Therefore, time dimension represents data in different snapshots. Three types of time stamps have been defined in (Bruckner and Tjoa, 2001) that are valid time, revelation time and load time.

Among the Real-Time Data Warehouse sources, new type of applications is contemporarily introduced that generate data streams (Babcock *et al*., 2002). The underlined issue in processing data streams is memory management due to large size of their continuous elements. To deal with this issue, approximations techniques are exploited (Widomet *et al*., 2003). Several approximations building methods include sampling, histograms and wavelets (Guha and Koudas, 2002). The congressional samples have been utilized for approximations in (Acharya *et al*., 1999). Special kind of query processing is necessitated to this special data. Therefore, continuous queries have been developed for data streams (Babu and Widom, 2001).

We proposed a framework for efficiently managing data streams in the Real-Time Data Warehouse (Majeed *et al*., 2010). Another work on data streams in data warehousing domain is demonstrated that uses Grid technology (Tho and Tjoa, 2006). The work in (Tho and Tjoa, 2004) employs message queues to securely amass the data streams. Like message queues, ETL queues have also been exercised for data streams storage (Karakasidis *et al*., 2005). The data streams have fast, continuous and time varying characteristics. It is necessary to resolve the bursts to synchronize with processing capability. The Token Bucket technique regulates such bursty data (Turner, 1986; Tenenbaum, 2003).

An algorithm is step by step description of the computer program. To assess the correctness of algorithm, algorithms analysis is performed (Cormen, 2001).

MATERIALS AND METHODS

Technical Representation of Data Streams Processor: In (Majeed *et al*., 2010), we proposed the architecture of

Data Streams Processor. As mentioned, bursts are initially received in memory after filtering through continuous query. It is important to control streams in memory and protect them from dropping. Other processes can manipulate on regulated streams from memory straightforwardly. The technical architecture of Data Streams Processor is depicted in Fig. 1.

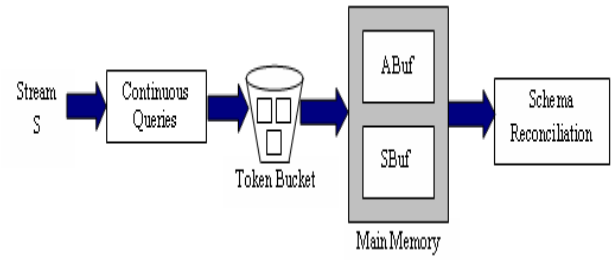

Fig. 1: Technical architecture of Data Streams Processor

According to architecture, data streams are driven by the source application to Data Streams Processor. These are filtered through the Continuous Query that throws irrelevant data streams. Afterward, bursts of data streams are regulated by the Token Bucket which transforms them into balanced shape. Continuous Query and Token Bucket are foundation to synchronize data streams rate with processor's service rate.

Significantly, memory is organized in a way that supports management of heavy data streams in limited capacity. It is partitioned into two buffers named Streams Buffer (*SBuf*) and Approximations Buffer (*ABuf*) that are discussed in detail in subsequent sections. The algorithm of approximations production uses these buffers for in-memory management of data streams.

Data Structures and Processes: The input stream is organized by different processes working on them. After processing, output stream is sent to ODS for joining with disk-based data and further transformations.

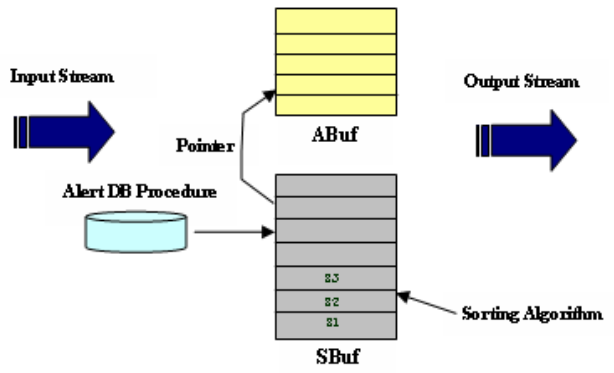

Fig. 2: Technical data structure and processes for Data Streams organization

Fig. 2 depicts the schematic diagram and data structure of approximations executed inside memory. As stated, data streams are accessed from the Data Source in burst form. To control their flow, bursts are regulated in balanced shape by Token Bucket method. Reader can clearly see the significance of Token Bucket positioned between the Continuous Query and memory. Where it provides regular shape to the traffic, it also saves streams from dropping as well support management to limited memory. According to Fig. 2, the available memory is partitioned into two buffers, the Streams Buffer (*SBuf*) and Approximations Buffer (*ABuf*). From now, we assume incoming streams as *S* and tuples in *S* as *w*. The *SBuf* stores incoming stream tuples *w*. In the other hand, *ABuf* is allocated for approximations storage generated from overloaded *w* tuples. A threshold *tlimit* is set in *SBuf* to signal overloading of *w* tuples. The threshold is set on memory according to percentage of memory decided by administrator like 80%. While a pointer *p* is being used to keep up the sequence of *S* streams for subsequent processing.

The processing of stream buffering is carried out with following steps:

(1) Once passing from the Token Bucket, *w* tuples of stream *S* are initially placed in *SBuf*.

(2) At some moment, *w* tuples exceed *tlimit* due to bursts of Stream *S* and simultaneously an alert triggers the approximations procedure.

(3) The approximations procedure then picks up a population *n* from overloaded *w* tuples to fabricate approximate answer.

(4) Right after taking population *n*, approximations procedure places a pointer *p* plus timestamp *t* in first slot of *SBuf* where first *w* tuple of population picked. The pointer *p* points to the first slot of available space in *ABuf* for residing approximations by procedure.

(5) Finally, sorting is applied periodically to maintain the ordering and utilizing free space.

The timestamp *t* associated with each tuple *w* and pointer *p* has been used to maintain the sequence of *w* tuples in *SBuf* for subsequent processing by the Data Streams Processor. Hence, sorting algorithm orders tuples *w* on the basis of timestamp *t*. It also eliminates free spaces generated as a result of First in First out (FIFO) approach adopted by the Data Streams Processor. This way, possible capacity is reserved in *SBuf* for new arriving tuples *w*. Using FIFO approach, incoming Streams *S* are added successive to last arrived tuple *w* whereas Streams Processor utilizes old most tuple *w* earliest.

The user demands for quick results to make up-to-date strategic decision making. Therefore, approximations play an important role in this situation. The data warehouse store summaries as well as detailed data. It provides results of queries on the basis of already

calculated summaries from detailed data with minimal delay. In addition, query profiles are also stored in the data warehouse. The query processor evaluates profiles and summaries for achieving fast results of a query.

Algorithm: Generate Approximations
Input: *t* tuples above *tlimit* in *SBuf*
Output: Approximated tuples in *ABuf*
Parameters: *tlimit* threshold in *SBuf*, *n* size of population
Method:

1. While (Length (*SBuf*) ≥ *tlimit*)
2. Read population *n* of *t* tuples from *SBuf*- above *tlimit*
3. Add pointer *p* in *SBuf* refer to population *ABuf*
4. Approximate (*n*, *ABuf*)
5. Add approximation result in *ABuf* pointed by *p*
6. End While

Fig. 3: Approximations algorithm

In Fig. 3, while loop in line 1 tests the threshold. It returns true as soon as data streams keep on exceeding from threshold. Length function with parameter *SBuf* count data streams currently in *SBuf*. When length of *SBuf* becomes less than *tlimit*, control exits from the loop. Data Streams Processor in line 2 reads population of *n* tuples from *SBuf*. In line 3, a pointer holding address of available free location in *ABuf* is positioned at the first slot of *SBuf* where population was picked. While line 4 contains approximations procedure that takes population of *n* tuples and *ABuf* as parameters. Finally, approximations procedure put results into *ABuf* in line 5.

Data Structure in action: The approximations are continually built after exceeding threshold point in memory. Fig. 4 demonstrates the production of approximations at different time instants.

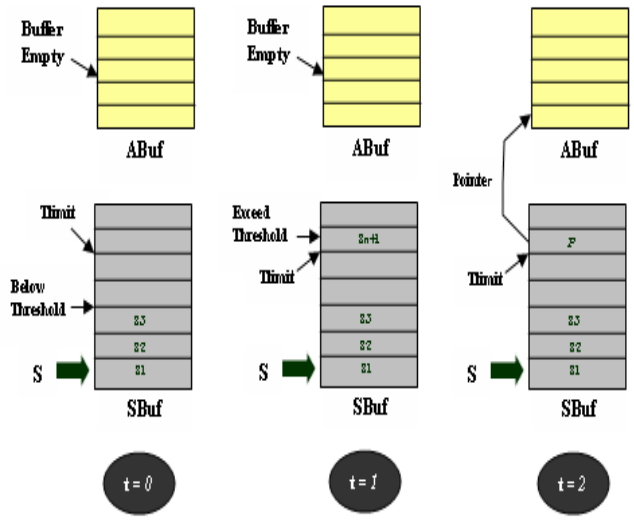

Fig. 4: Execution of memory processes in different intervals

The operation of approximations algorithm in different instants is illustrated in detail. This can be described in following steps.

(1) At time *t=0*, *w'*s from *S* are stored in *SBuf*. *ABuf* is empty this instant.

(2) At time *t=1*, *w'*s exceed *tlimit* in *SBuf*. Instantly approximations alert triggers the approximations procedure.

(3) At time *t=2*, Approximations procedure picks up *n w'*s from *SBuf*.

The concept of approximations has been clarified in Fig. 4. It is worth claiming that logical partitions of memory into separate buffers eliminate ambiguities. Hence memory is used optimizely and resolves the issue of limited memory.

The Token Bucket sends *w'*s towards memory in regular rate. These *w'*s in turn are used by Data Streams Processor with comparable rate. As a result, Data Streams Processor harmonizes with the pace of incoming streams. Moreover, incoming streams are reasonably consumed by limited memory and avoided from descend.

The *w'*s whose processing is completed, leaves the memory and transmitted to ODS for join with disk relations and further transformations. The approximations procedure continues until exceeding *w'*s reach below *tlimit*.

RESULTS AND DISCUSSION

In this section, we are applying a case study (Raden, 2003) on the proposed real time data warehouse architecture. The Gately Company has following analytical needs to perform on their real-time data warehouse (The quoted paragraph below states the requirements that should be fulfilled from real-time data warehouse).

"Gately is a chain of web stores, selling affordable products with high-quality, service and integrity. Their primary objective is to obtain numerical data on the return on investment of each of their internet advertisements, especially their Google Adwords advertisements, to determine which ads were costing more than they brought in. A secondary objective was to create a system which could report which items were ordered through the website, and how often, and how much was spent on them".

Gately website generates data streams maintained in the server log. They have great number of customers buy their products. The customers visit Gately website for purchasing offered products. As they visit the website through Adwords or directly, the information of their clicking history like ordered products etc. are stored in web server log. This information is required by the RTDW for analysis demanded by the company. A sample log is demonstrated in Table 1:

Table 1: Web server log file records

Source of Request (Host)	Bytes	Referring Page	Date and Time of Request Browser	Page Requested (HTTP protocol) Platform	PID	OID	CID
Pm471-46.dialip.mich.net	1449	"http://www.gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:44 -0400]	"GET /images/tagline.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	10659	"http://www.gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:44 -0400]	"GET /images/bkgrnd.jpg HTTP/1.0"	P131	O142	C101
Pm471-46.dialip.mich.net	280	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:44 -0400]	"GET /images/yellow_bit.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	1292	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	24/May/2009:19:13:44 -0400]	"GET /images/TE_logo.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	714	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:52 -0400]	"GET /images/site_map.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	43	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:53 -0400]	"GET /images/home_00.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	43	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:53 -0400]	"GET /images/use_eval_hbut.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	747	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:55 -0400]	"GET /images/use_eval_hbut.gif HTTP/1.0"	P131		C101

Table 2: Discarded records by continuous query

Source of Request (Host)	Bytes	Referring Page	Date and Time of Request Browser	Page Requested (HTTP protocol) Platform	PID	OID	CID
Pm471-46.dialip.mich.net	43	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:53 -0400]	"GET /images/home_00.gif HTTP/1.0"	P131		C101
Pm471-46.dialip.mich.net	43	"http://www. gately.com/" "Mozilla/4.51 [en] (Win98; I)"	[24/May/2009:19:13:53 -0400]	"GET /images/use_eval_hbut.gif HTTP/1.0"	P131		C101

Such records in the log file are created in huge quantity in a unit time *t*. We are going to apply the sample log file on each component of Data Streams Processor.

Continuous queries: Click streams are reached to the Data Streams Processor. On first step, the Data Streams Processor passes them through continuous query. The Continuous query here is used to filter click streams. It discards irrelevant data streams and pass only required data streams to the next process. For example, if click on a link by the customer generates number of bytes less than 50, it means he/she does not perform any substantial activity. The query in Fig. 5 discards such records from coming data streams:

```
SELECT *
FROM LoginInfo L
WHERE L.bytes < 50
```

Fig. 5: Continuous Query

So the following records of sample log file presented in Table 2 are discarded by this continuous query.

Token Bucket: Before applying any further process, the irregular streams are necessary to be converted into a regular flow so that the Data Streams Processor can easily synchronize with the pace of data streams. The token bucket is used in Data Streams Processor to convert data streams into a regular flow and smooth out bursts. In the algorithm of Token Bucket, the bucket holds tokens generated on clock ticks with the rate of one token every ΔT sec. With each token, a specified number of streams, say 10, are flowed for further processing. The algorithm is implemented in following way:

A variable is used as a counter which is incremented on each clock tick and decremented on transmitting specified number of data streams against each token. The result of using this technique in the architecture is regular flow of data streams.

Let data streaming application sends a burst of 5000 records per second in particular interval. Whereas the Data Streams Processor is capable of processing 3000 records per second. In this situation, Token Bucket increases passing of streams per token to approximate output rate with input rate. Consequently, burst is smoothes out and memory processes take enough time to manage streams.

Data Streams Approximations: After passing from the continuous query and the token bucket, click streams are required to be stored in memory. As discussed in earlier sections that memory cannot keep all data streams at each time interval. The data streams are frequently generated in burst form. According to the RTDW architecture, increasing rate of data streams from specified threshold of memory is approximated. It use approximations algorithm for summary results based on an efficient sampling technique.

Assume that click streams in memory are exceeding from the specified threshold limit. The extra records are 1000 from which a sample of 100 records is taken. The continuous query to generate approximate results is demonstrated in Fig. 6:

```
SAMPLE 100 OF
SELECT*
FROM LogInfo
```

Fig. 6: Query for approximations

It randomly selects a sample of 100 records from the population of 1000 records.

Table 3: Order of columns in log file

Source of Request (Host)	Bytes	Referring Page	Date and Time of Request Browser	Page Requested (HTTP protocol) Platform	PID	OID	CID

The order of columns after conversion relevant to target table:

Table 4: Order of columns in ODS

OID	P ID	C ID	Source of Request (Host)	Bytes	Referring Page	Date and Time of Request Browser	Page Requested (HTTP protocol) Platform

Schema Reconciliation: The next step is to change format of click streams relevant to the format of ODS. Log file in Table 1 is in relational format. It is necessary to convert that in the format of target schema. Frequently ODS is implemented in relational structure. The degree, order of columns and field lengths etc. are necessary to be changed according to the target tables. For Example, The current order of columns in the log file is:

The columns PID, OID and CID in Table 3 are switched at beginning in Table 4.

Acknowledgement: This article is extended part of Fiaz Majeed's (author) MS thesis done under the supervision of Mr. Sohaib Mahmood in CIIT Lahore, Pakistan. A paper titled "Efficient Data Streams Processing in the Real-Time Data Warehouse" has already been published in IEEE ICCSIT.

Conclusion: In this work, framework of real time data warehouse is technically analyzed. The theoretical foundations laid in the work (Majeed *et al*., 2010) are augmented on technical bases. For this, in-memory approximations are discussed with sufficient detail. During annotation of approximations algorithm, the operation of approximations process in memory at different time instants is also clarified. Moreover, the technical data structure for approximations production is discussed for implementation point of view.

REFERENCES

Acharya, S., P. B. Gibbons, and V. Poosala, Congressional samples for approximate answering of group by queries, In Proceedings of the special interest group on management of data, 487-498 (1999).

Babcock, B., S. Babu, M. Datar, R. Motwani, and J. Widom, Models and issues in data stream systems, In Proceedings of the 2002 ACM Symp on Principles of Database Systems, 1-16 (2002).

Babu, S., and J. Widom, Continuous queries over data streams, SIGMOD Record, 30(3): 109-120 (2001).

Basu, R. Challenges of real-time data warehousing, Information Management Special Reports (2003).

Brobst, S. Delivery of extreme data freshness with active data warehousing, Journal of data warehousing, 7(2): 4-9 (2002).

Bruckner, R., and A. M. Tjoa, Managing time consistency for active data warehouse environments, In Proceedings of the 3rd intl. conference on data warehousing and knowledge discovery, 254-263 (2001).

Cormen, T. H. Introduction of Algorithms, Second Edition, McGraw-Hill (2001).

Guha, S., and N. Koudas, Approximating a data stream for querying and estimation: Algorithms and performance evaluation, In Proceedings of the data engineering, 567-576 (2002).

Inmon, W.H. Building the data warehouse. New York: Wiley (1996).

Karakasidis, A., P. Vassiliadis, and E. Pitoura, ETL queues for active data warehousing, In Proceedings of the Annual MIT IQ Industry Symposium (IQIS), 28-39 (2005).

Majeed, F., M. S. Sohaib, and M. Iqbal, Efficient Data Streams Processing in the Real-Time Data Warehouse, In Proceedings of 3rd IEEE Intl. Conference on Computer Science and Information Technology (ICCSIT), 57-61 (2010).

Mohanty, M. Data Warehousing: Design, development and best practices, New Delhi: Mcgraw hill (2006).

Motwani, R., J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, Query Processing, approximation, and resource management in a data stream management system, In Proceedings of the Conference on Innovative Data Systems Research (2003).

Ponniah, P. Data warehousing fundamentals, New Jersey: Edison (2001).

Raden, N. Exploring the business imperative of real-time analytics, Teradata White Paper, 1-14 (2003).

Tho, N. M., and A. M. Tjoa, Zero latency data warehousing for heterogeneous data sources and continuous data streams, In Proceedings of the 5th intl. conference on information integration, web applications and services, Jakarta, Indonesia (2004).

Tho, N. M., and A. M. Tjoa, Zero-latency data warehousing (ZLDWH): the state-of-the-art and experimental implementation approaches, In Proceedings of 4th IEEE Intl. conference on computer science research; innovation and vision for the future (RIVF), 166-175 (2006).

Tenenbaum, A. S. Computer Networks, Fourth edition, Prentice Hall PTR (2003).

Turner, J. S. New directions in communications (or which way to the information age), IEEE Commun. Magazine, 24: 8-15 (1986).

Vandermay, J. Considerations for building a real-time data warehouse, Data-Miror Corporation White Paper (2000).