Vol 02, Issue 02, June, 2011 Research Journal of Computer Systems Engineering- An International Journal
http://technicaljournals.org ISSN: 2230-8563; e-ISSN-2230-8571

A review on Processing of Data Streams

[1]P. Arumainayagam, [2]E.Y.A Charles and ,[3]S. R. Kodituwakku
[1]Computer Unit, University of Jaffna, Sri Lanka
[2]Department of Computer Science, University of Jaffna, Sri Lanka
[3]Department of Statistics & Computer Science, University of Peradeniya, Sri Lanka.
{ uojtheepa@gmail.com, Charles.ey@jfn.ac.lk, salukak@pdn.ac.lk }

Abstract
In a database management system (DBMS), data are stored and processed whenever necessary. On the other hand in data streams, data are created and processed every millisecond. Due to some characteristics of data streams, processing of them is a challenging task. This paper reviews data streams, their characteristics and the challenges to be faced in processing them.

1. Introduction

In data streams, vast amount of data are entered into the system in each and every millisecond. So while processing the data streams huge amount of data is to be handled. Querying over these data is also a difficult task. Data in a database management system (DBMS) can be processed using Structured Query Language (SQL). But data streams can't be processed efficiently using this language. Variety of languages for processing data streams are introduced [1], [2], [3], [6], [7] to avoid the problems in accessing data streams using Structured Query Language (SQL). Keeping the old data while new records arrive is another problem. As vast amount of data arrive in every millisecond, saving of all these data in a data stream management system is impossible. So the older data have to be either deleted or kept in any other format. Language proposed in [2] deletes old data or replaces old data by newly arrived data. These two ways are very easy, but problems arise when queries need to access old data to produce outputs. Three important facts have to be considered to get better output from stream processing. They are: less memory requirement, minimize the tuple latency and accuracy of produced answer. Memory requirement is an important fact to be considered when creating a data stream management system, because bulk amount of data have to be saved in each and every millisecond. So memory is to be handled very carefully and efficiently. In real time systems, users/customers will be in waiting for the answer while data stream processing is going on. So tuple latency will be reduced very much and the answer is to be given to the users as early as possible. This implies that very little tuple latency will lead to a very efficient system. Another fact to be considered is the accuracy of the answer. Sometimes two or three tuples have to be dropped among that bulk of tuples. This may lead to inaccuracy of answers. In [4], Quality of Service (QoS) of the system is described. The QoS is determined by these dropped tuples. If the accuracy of the answer is increased then QoS of the system is also increased. So these dropouts have to be avoided. Otherwise the answers have to be given to the users with the percentage of accuracy. Better memory management, very small tuple latency and much accuracy of answers in processing data stream will lead to an efficient system.

2. Data Streams Vs Databases

A finite set of data is stored in a database. Then these data can be queried using SQL whenever necessary. But in data stream management system, data are to be continuous. It will enter into the system continuously. Querying over these data stream is a very difficult and challenging task. SQL can't be used to querying over these streams, because this language has no features to handle data streams. Therefore, SQL is insufficient for processing data streams. To process these types of queries Continuous Query Languages are used. These types of languages use high power of SQL. While querying over the data streams, not only the newly arrived streams but also the streams resulted from queries or sub queries also be considered. Major differences between DBMS and data Stream Management Systems (DSMS) are as follows:

DBMS	DSMS
Persistent relations	Relatively low update rate
One-time queries	Transient streams
Random access	Continuous queries
Unbounded disk store	Sequential access
Only current state matters	Bounded main memory
Passive repository	History or arrival order is critical
Active stores	No real time services
Possibly multi GB arrival rate	Data stale or precise
Real time requirements	Unpredictable or variable data arrival and characteristics

Table 1: DBMS VS DSMS

If the processing of data stream is compared with processing of database, there exist some problems due to the following characteristics of data streams.

1. Operates on high volume of data.
2. Queries are long-run and continuous.
3. Have to make near real-time decisions.

In databases, data are stored in the database and processed later whenever needed. In case of data streams, better result will be provided with negotiable tuple latency when they are processed on-the-fly, i.e., before they saved into the data stream management systems. This technique will lead to an efficient system. The arriving streams are temporarily stored into the queues before they are queried. Scheduling strategies used in [9] schedule these tuples from queues to queries. Those scheduling algorithms are mainly focused on the memory minimization. Two scheduling strategies as path capacity strategy and chain scheduling strategy are analyzed here. As the result of analysis these two strategies, new strategies are introduced as segment scheduling strategy and threshold strategy which overcomes the existing problems. Better tuple latency is achieved by path capacity and better memory requirement is achieved by segment scheduling. The newly created strategies met acceptable performance but they didn't meet all of the following characteristics:

1. With fixed amount of resource, reach maximum performance.
2. Handle the overload problem with efficiency.
3. Guarantee the required Quality of Service for queries.
4. Easily implemented, efficient running with minimal overhead.

With limited resources, it is impossible to satisfy these requirements. Scheduling strategies introduced in [9] also do not satisfy all these important characters. But all of these characters have to be considered when design an algorithm for scheduling. This will lead to an efficient system. Some performance metrics discussed in [9] includes:

1. Tuple latency and throughput: - In the term of tuple latency path capacity scheduling shows better results than others.
2. Memory requirement: - In the case of memory requirement both path capacity strategy and chain scheduling strategy have an optimal property. If the system is not overloaded both strategy are similar in case of memory requirement.
3. Starvation problem: - Under some constraints both strategies have starvation problem.
4. Scheduling overhead: - This problem also exists in both strategies. But under some circumstances path capacity strategy has less scheduling overhead than chain scheduling.
5. Context switching overhead: - One of the very important fact to be considered which influence on the performance of the system.

Existing algorithms show good results for some of the performance metrics while showing bad results for the rest of the metrics. So when design an algorithm for scheduling, better output for these performance metrics should also be considered. That means new algorithms need to produce good results for all of these performance metrics and the results must be better than the existing algorithms.

3. Characteristics of data streams

Data stream will change in quality and value with time. Temperature, pressure, traffic, ATM transactions, on-line mining of web usage, phone calls and some others can be considered as example for data streams. All of them have the above mentioned characteristics. This is not the only one characteristic of data stream but there exist some more characteristics. In [1], eight requirements are given which are to be considered while processing the data streams. They are:

1. Keep the data moving while the process time. This will reduce the latency, because storing the data before processing will lead to unwanted tuple latency. This will reduce the performance of the system.
2. Language for the stream processing is the second requirement. SQL is widely used for database processing. But for processing data streams this is not an efficient language. Some continuous query

Vol 02, Issue 02, June, 2011 Research Journal of Computer Systems Engineering- An International Journal
http://technicaljournals.org ISSN: 2230-8563; e-ISSN-2230-8571

languages exist to process these streams. In [1], StreamSQL which is an extended version of SQL is mentioned as language for processing stream.

3. Handling the imperfection data such as lost data, delayed data or out-of-order data. Such data will lead to wait the system indefinitely for producing the answer. To solve this problem, continuous processing with partial data is proposed as a solution [1]. But this will cause the problem of inaccuracy. Therefore, this problem needs to be solved or the answers have to be provided with the percentage of accuracy.
4. Predictable and repeatable outcomes. By this property the results of the processing can be decided.
5. Storing of data in an efficient manner. First advantage of this requirement is the efficient memory usage. At some query processing, older data have to be compared with newly arrived ones. Careful data management will lead to efficient processing of these types of queries. Additionally tuple latency will also be reduced.
6. Availability of the system needs to be ensured when failure occurs. To get high performance and high accuracy of answers, the data in the system have to remain safely in the failure time and also be able to recover from the system. Another problem at failure time of stream is losing of newly arrived data. If we consider the database, at the failure of the system, there is no data arrival. But in the case of data streams data arrive continuously. That means data would be arrived at the system failure time and they have to be retrieved after the recovery of the system.
7. Ability of the system to work in a distributed environment. By this capacity of the system, it can get the usage of multi processors and can get the high performance.
8. Ability of the system to process lots of data in a negotiable time. This will lead to very low tuple latency.

4. Challenges in processing streams

As a bulk amount of data is created in each and every millisecond, they have to be processed without delay. This is not a simple matter. Processing, storing and retrieving these streams are subject to a lot of challenges. They are discussed in this section. Some challenges are discussed in [2]. They are,

1. Input records have to be accessed sequentially.
2. Some additional states have to be stored with these streams.
3. Some operators are blocked. They may produce null outputs as streams are infinite.
4. Streams have to be accessed before storing them and outputs have to be produced continuously.

While creating a data stream Management system these challenges have to be faced. Also they have to be won. While accessing big amount of data in each and every millisecond, non sequential access of data cause some problems. In some cases, data may be given priority to access. Here high priority data get chance to process while low priority data wait for long time. For every second amount of data will be increased indefinitely. In this situation these waiting data may have to wait indefinitely for processing. At online access of these data this is a very big overhead to the system. With the stream data, some other additional but important facts also may have to be considered. For example, the time of the stream's arrival will be useful for lot of future processing of those streams. Accessing these data streams before storing them will lead to very less tuple latency. Storing the data first and then taking them again to process also take time which will lead to tuple latency. These are not the only challenges. There exist lots of other challenges. In [5], two more challenges are described in building an efficient DSMS. They are:

1. Rather than managing stored data in traditional databases, here multiple continuous, unbounded, rapid and time- varying data streams are handled and managed.
2. Because of the continuous nature of data, the output for the long-running queries is expected to be continuous and timely fashion.

With these challenges, in [9], some more problems in processing data streams are mentioned as,

1. Operates on high volume of data
2. Queries are long run and continuous.
3. Have to make near real time decisions.

When a huge amount of data arrives within very small time intervals, managing those data is very challenging task. Which data is to be accessed first, which data are to be sent first to the queries, all of these are the challenging facts. In DBMS, one query has a finite number of answers. Here data arrives continuously; this implies answer to the queries also continuous. For example, in phone call records, find the average amount of phone calls for each five minutes. The answer must be produced for each and every five minute. The answer is continuous for every five minutes. Produced answer will be changed with time because of the change of data. In some situations, the answers of the queries have to be predictable. So some real-time decision, which will be true, has to be taken at processing of data streams. Another big challenge in stream processing is the storing of this huge amount of data. With the limited amount of memory, storing all of these data is impossible. Some system deletes the older data when new data arrives. But this will cause lot of operational problems. Some

Vol 02, Issue 02, June, 2011 Research Journal of Computer Systems Engineering- An International Journal
http://technicaljournals.org ISSN: 2230-8563; e-ISSN-2230-8571

operations may need older data to complete its processing and producing the answer. So deleting the older data is not an efficient way. Older data have to be saved in the system in any format which will not affect the memory.

5. Language for processing Streams

SQL, which is a widely used language for processing databases, is not an efficient language for processing data streams. It has no capacity to efficiently process data streams. Lot of stream query languages is used for data stream processing. In [1], StreamSQL is used as a stream processing language. This language is an extended version of SQL, which contains special features to process streams. In [2] and [3], Continuous Query Language which is very close to SQL is introduced to process streams. This uses the high power of SQL. Condensative Stream Query Language (CSQL) is introduced in [6] for stream processing. This language is more expressive than others. Another language is discussed in [6] as StreQuel. Here each query definition is followed by a for loop construct. The Stream Mill system descried in [7] supports the Expressive Stream Language (ESL) which has the superior expressive power than any other stream language.

6. Conclusion

Processing of data streams and the challenges in processing such streams are described in this paper. After the analysis, some important facts are noticed while creating the new data stream management systems. First one is the accuracy of the system. We discussed how drop of tuples will lead to the inaccuracy of answers. To reduce this inaccuracy, the dropouts have to be minimized. The data streams have a characteristic of predicting future values. Using this property, the dropped tuples can be found and processed to find more accurate answers. Another problem is the difficulty in storing of huge amount of data. This problem is solved in the existing systems by replacing the old data by new ones. This leads to inaccessibility of old data. The older data need to be preserved in order to solve this problem. This can be resolved by entering the number of occurrences of each data rather than repeated entering of the data. This will utilize the memory usage. With the number of occurrences, time of the data arrivals may also be saved for the future use. The time information may be the additional information saved with the record. The other major problem to be considered is the system failure time. The data arrived at the system failure time are to be retrieved after the system recovery. This is a critical problem. At this time the data may be saved at another system or may be taken by the prediction character of the streams. If the data is saved in any other system, those data may be taken from that system and processed after the system recovery. More than one language is discussed here for processing the streaming data. Each of them has their own advantages and drawbacks. So a suitable language among those languages needs to be found in order to process the stream.

7. References

[1]. Erik Zeilter, Tore Risch. ,Scalable Splitting of Massive Data Streams , Department of Information Technology**,** *DASFAA 2010***,** PartII, LNCS 5982, pp 184-198,2010.

[2]. Jurgen Kramer. , Continuous Queries over Data Streams – Semantics and Implementation, *BTW 2009***:** 438-448.

[3]. Michael Stonebraker, Ugur Centintemel, Stan Zdonic. , The 8 Requirements of Real-Time Stream Processing, Department of Computer Science, *SIGMOD Rec.***,** Vol. 34, No. 4. (December 2005), pp. 42-47.

[4]. Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Wisdom, STREAM: The Stanford Data Stream Management System, *SIGMOD Conference 2003***:** 337-348.

[5]. Badrish Chandramouli, Mohamed Ali, Jonathan Goldstein, Beysim Sezgin, Balan S. Raman , Data Stream Management Systems for Computational Finance, *Computer***,** vol. 43, no. 12, pp. 45-52, Dec. 2010

[6]. Hetal Thakkar, Barzan Mozafari,Carlo Zaniolo, Designing an Inductive Data Stream Management System: the Stream Mill Experience, *SSPS '08 Proceedings of the 2nd international workshop on Scalable stream processing system***,** ISBN: 978-159593-963-0**.**

[7]. Qingchun Jiang, Sharma Chakravarthy, Scheduling Strategies for a Data Stream Management System, Computer Science & Engineering, *BNCOD 2004*:16-30

[8]. Lewis Girod, Yuan Mei, Ryan Newton, Stanislav Rost, Arvind Thiagarajan, Hari Balakrishnan, Samuel Madden, XStream: a Signal-Oriented Data Stream Management System., *In Proceedings of the International Conference on Data Engineering (ICDE'08), Cancun, Mexico, April 2008*.

[9]. Daniel J.Abadi, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, Stan Zdonik, Aurora: a new model and architecture for data stream management, *The VLDB Journal — The International Journal on Very Large Data Bases archive* Volume 12 Issue 2, August 2003**.**

Authors Biography:

Miss. P. Arumainayagam received the B.Sc. degree from University of Jaffna, Sri Lanka, in 2008. From 2008 August-2009 August she worked as Instructor (On Contract) and from 2009 September to 2010 September she worked as an Asst. Lecturer (On contract) in Department of Computer Science, University of Jaffna. From October 2010 to till date she is working as Instructor in Computer Technology at Computer Unit, University of Jaffna. Now she is an M.Phil student at University of Peradeniya, Sri Lanka.

Dr. Eugene Y A Charles studied at the University of Jaffna, Sri Lanka where he received his B.Sc Degree in Computer Science. He obtained his PhD at the Cardiff University, UK. At present, he is a senior lecturer and Head of the department of Computer Science, University of Jaffna. His main research interests are focused on data mining, machine learning and neural networks.

Saluka Ranasinghe Kodituwakku is an associate professor at the Statistics and Computer Science Department, University of Peradeniya, Sri Lanka. His research interests are database systems, distributed computing and software engineering.