Experimental Analysis on Processing of Unbounded Data

Nirav Bhatt, Amit Thakkar

Abstract: ***Processing of unordered and unbounded data is the prime requirement of the current businesses. Large amount of rapidly generated data demands the processing of the same without the storage and as per the timestamp associated with it. It is difficult to process these unbounded data with batch engine as the existing batch systems suffer from the delay intrinsic by accumulating entire incoming records in a group prior to process it. However windowing can be useful when dealing with unbounded data which pieces up a dataset into fixed chunks for processing with repeated runs of batch engine. Contrast to batch processing, stream handling system aims to process information that is gathered in a little timeframe. In this way, stream data processing ought to be coordinated with the flow of data. In the real world the event time is always skewed with the processing time which introduce issues of delay and completeness in incoming stream of data. In this paper, we presented the analysis on the watermark and trigger approach which can be used to manage these unconventional desires in the processing of unbounded data.***

Index Terms: Unbounded Data, Window, Trigger, Watermark, Stream Data.

I. INTRODUCTION

Due to growth of the social media and telecommunication technologies, large amount of data is generated continuously and at high rates. Such data is called stream data. It is not practically possible to store such data physically and standard data-mining techniques cannot handle it. So system or model that analyzes and manages stream data is are called stream management system [12]. Here some kind of approximation is required as models cannot observe and compute the entire data exactly [3]. Stream data processing is crucial at the same time complex in big data environment and the causes are:

- Businesses process the data having particular time stamp and need to process in time order. Moving to stream processing is a better way to deal with the issue of latency.
- Huge and infinite data which are progressively usual in recent trade are efficiently controlled with a specific system which is intended for such continuously arriving data.
- Processing data in such system is challenging task and what we require is type of processing mechanism which is well defined by considering processing of continuously arriving data.

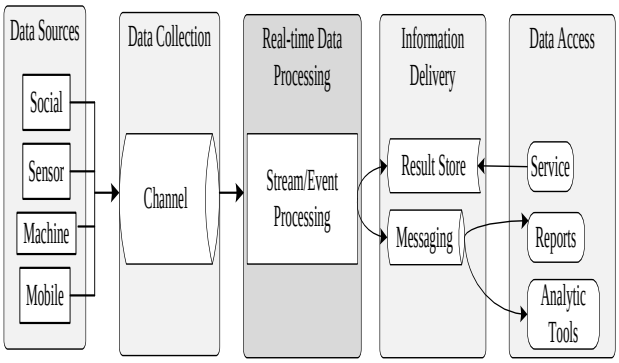

Fig 1. Stream Processing Framework

Above fig.1 shows structure of stream processing. The structure evaluates either single component or a compact window with current data. Computation in stream processing is instantly and taking very few seconds. Stream processing technique is capable to generate almost real time data since it processes the data as soon as arrives. A very high level of precision is present in such system, which is important for processing temporal data. Many popular tools such as Storm is a critical infrastructure that powers many of the real-time data-driven decisions made so far [13].

II. PROCESSING OF UNBOUNDED DATA SETS

In case of batch processing system, it is possible to define the operations on entire dataset but in case of stream processing system, operations have to apply to an individual dataset. These datasets in stream processing are considered as unbounded datasets. Unbounded data processing have following two different time domains.

- Event-time: "which is the time at which events actually occurred".
- Processing-time: "which is the time at which events are observed in the system".

Ideally, event-time and processing-time should be identical, as event being considered to be processed as soon as they happen [10]. However in the real world situation, event time is always skewed with the processing time which is quite variable and rely on the features of the processing engine, hardware and also the input provided [1,16]

Revised Manuscript Received on July 05, 2019.

Nirav Bhatt, Department of Information Technology, CSPIT, CHARUSAT, Changa, Anand, Gujarat, India. E-mail: niravbhatt.it@charusat.ac.in

Dr. Amit Thakkar, Department of Information Technology, CSPIT, CHARUSAT, Changa, Anand, Gujarat, India.

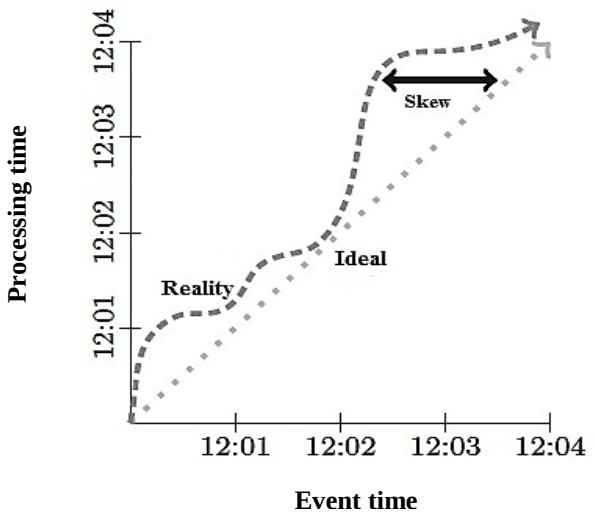

FIG 2. Event-Time vs. Processing-Time

III. WINDOWING

On the arrival of new data, system should be able to cope with it and old data may be updated. The most widely recognized approach to process an unbounded data is with the frequent execution on batch engine. The received data is transformed into fixed small size windows and where every window is processed separately [1].

Objective of Windowing is to convert an input data source into predetermined pieces for processing. Below figure-3 express the various windowing mechanism [1, 7].

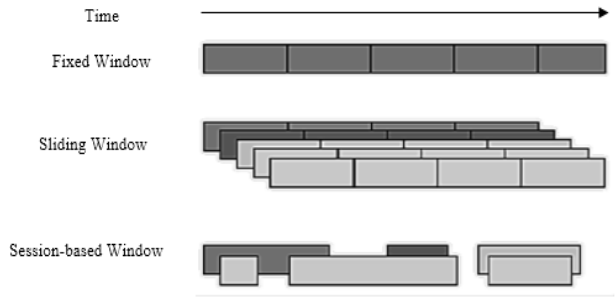

Fig 3. Types of windowing strategies

In a Fixed window based approach, the whole dataset is distributed and organized into fixed size window however sliding window has fixed size and span that slides across the data where size indicates the length of each window, and a slide determines the interval between two consecutive windows [15]. Although there are some novel windowing approach has been introduced which is used based on the application such as count Windows define the size and the slide based on number of elements [15]. If the span is less than the size, then the window overlap [1]. Session window is known as dynamic window. There are sequences of events ended with the gap of idleness which may be larger than timeout.

Windowing can be categorized into following two categories,

1) Processing time window
2) Event time window

A. Processing time window

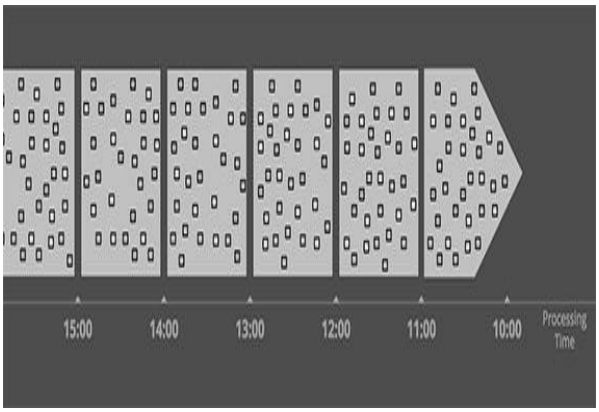

Fig 4. Fixed windows by processing time [1]

In processing time window, incoming data are gathered into the fixed windows as per the sequence they reach into the pipeline. The system store the data received into windows up to the time it required to process. For instance, consider the fixed windows of ten minutes, the system will integrate data for ten minutes of processing time, and then it will consider the collected data in those ten minutes as a window and process them [1]. Processing time semantics are always non-derivable, as the performance of the framework relies upon the speed at which data enter the system and on the speed at which the data flow among different operators within the engine [15].

B. Event time window

In event time window, incoming data are gathered into the fixed windows as per the times they occurred, regardless of when and in what order records arrive, process the data as per an event timestamp associated with them. still there are two shortcomings of such windowing system: 1) Latency: in event time based windowing system, there is delay in processing as data with specific time stamp may arrives late and need to be consider in the processed window. 2) Completeness: as the late data causes the delay in processing, it also be the reason for not identifying the end of input for current window.

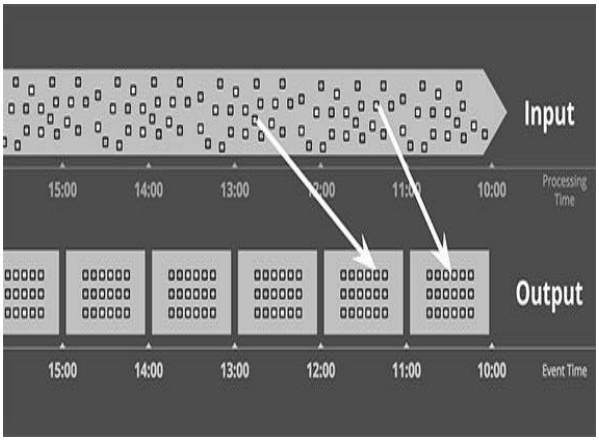

Fig 5. Fixed windows by event time [1]

However discretized streams (D-Streams) can be used to treat streaming as a series of short batch jobs, and bring down the latency of these jobs as much as possible [14]. Following table-1 shows the comparative analysis of distinct tools and techniques to handle the issues of latency and throughput.

Table 1. Comparative analysis of tools and techniques to deal with issues of event-time window

Tools / Techniques	Approach to deal with Latency & Throughput
Apache Strom & Flink [8]	Count Window, Trigger
Stream Box [7]	Epoch & Container System
Apache Flink [4]	Buffering mechanism
DSP Engine [6]	Queue of Events before processing
Google Data Flow [5]	Watermark & Trigger

IV. WATERMARK & TRIGGER

In a stream of events, Watermarks are used to measure the progress and completeness relative to the event times of the records. It is function defined by,

$$F(Pt) \rightarrow Et \qquad (1)$$

Above equation (1) defines that the point in event time Et is the point up to which the system believes that all inputs with event times less than Et have been observed and no more data with event times less than Et will ever be seen again [2,11].

A. Example of Watermark

When a watermark is received, all windows up to that timestamp will be evaluated. For example,
Consider the Window length = 20s.

Events e1(5:59:55), e2(6:00:05), e3(6:00:17), e4(6:00:28), e5(6:00:36) are received with defined watermark w1 = 6:00:30.

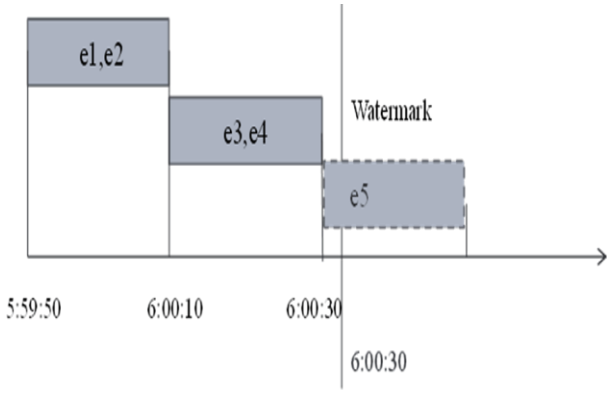

Fig 6. Example of watermark in stream data processing

Here, e5 is not evaluated since watermark timestamp 6:00:30 is lesser than the event ts 6:00:36.

Triggers declare when output for a window should happen in processing time. Each specific output for a window is referred to as a pane of the window [2]. Triggers can be of different types such as event-time trigger, processing-time trigger and count trigger.

B. Analysis of windowing and triggering approach on different stream processing tools

Following Table 2 shows the support of different types of window and trigger mechanism with popular stream processing engine.

Table 2. Analysis of window and trigger on Implementation Tools

		Beam [9]	Google Cloud Data flow	Apache Flink	Apache Spark	Apache Hadoop MapReduce
Window	Fixed Window	Yes	Yes	Yes	Yes	Yes
	Sliding Window	Yes	Yes	Yes	Yes	Yes
	Session Window	Yes	Yes	Yes	Yes	Yes
Trigger	Event-time Triggers	Yes	Yes	Yes	No	No
	Processing-time Triggers	Yes	Yes	Yes	Yes	No
	Count Triggers	Yes	Yes	Yes	No	No

V. EXPERIMENTS

Experiments has been carried on incoming stream of user's score from distributed environment. Apache Beam is used to perform the experiments which support the different windowing and triggering mechanism. Following result is using Dataflow runner from the tool Apache Beam [17] [18] which performs the steam processing on incoming user data and provide the sum of user score periodically. Following shows the experimental analysis on apache beam for incoming stream of sum of user score data.

Input Dataset:

Input dataset contain scores of different users (user1, user2, user3) along with its timestamp values:

Format of dataset is as follow:

Input dataset contains "Username", "team name", "score", "time_stamp in milliseconds" for a gamming score of three different users.

Table 3. Experiments on "User Score"

	User Score
Input	Format: Username, team name, score, time_stamp_in_ms user1_A,A,10,1445230923951,2015-11-02 09:09:32.226 user2_B,B,10,1445230923951,2015-11-02 09:09:28.224 user2_B,B,10,1445230923951,2015-11-02 09:09:30.225 user3_C,C,10,1445230923951,2015-11-02 09:09:34.227 user3_C,C,10,1445230923951,2015-11-02 09:09:36.228 user3_C,C,10,1445230923951,2015-11-02 09:09:38.229 . . .
Output	user: user2_B, total_score: 20 user: user1_A, total_score: 10 user: user3_C, total_score: 30 . . .

VI. CONCLUSION AND FUTURE SCOPE

This paper has highlighted the processing of unbounded data sets. We have presented that how different types of windowing specifically event-time based and processing-time based is useful to process the continuously arriving data. We define the importance of watermark along with different triggering approach which is used to evaluate the progress and completeness in unbounded data processing. Experiments presented here shows the three different user's total score in gaming example which is gathered from distributed environment with varying timestamp.

The processing of unbounded data of user's score can further be improved by enhancing the basic pipeline of apache beam. The same pipeline can be executed on different runner like Apache Flink and apache spark supported by apache beam and hence the performance can be compared for different runner.

REFERENCES

1. Akidau, T. (2016). The world beyond batch: Streaming 101. Oreilly. com, 20. Available: https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
2. Akidau, T. (2016). The world beyond batch: Streaming 102. Oreilly. com, 20. Available: https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102
3. R. de Amorim, "Pascal Poncelet, Florent Masseglia, Maguelonne Teisseire: Successes and New Directions in Data Mining", *Information Retrieval*, vol. 12, no. 4, pp. 504-508, 2008.
4. Carbone et al., "Apache flink: Stream and batch processing in a single engine", Asterios.katsifodimos.com, 2015. [Online]. Available: http://asterios.katsifodimos.com/assets/publications/flink-deb.pdf.
5. T. Akidau et al., "The dataflow model", Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1792-1803, 2015. Available: 10.14778/2824032.2824076. Available: 10.14778/2824032.2824076.
6. J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl, "Benchmarking Distributed Stream Data Processing Systems," 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018
7. H. Miao, H. Park, M. Jeon, G. Pekhimenko, K. McKinley and F. Lin, "StreamBox: modern stream processing on a multicore machine", Dl.acm.org, 2017. [Online]. Available: https://dl.acm.org/citation.cfm?id=3154749.

8. L. Affetti, R. Tommasini, A. Margara, G. Cugola and E. Della Valle, "Defining the execution semantics of stream processing engines", Journal of Big Data, vol. 4, no. 1, 2017. Available: 10.1186/s40537-017-0072-9.
9. S. Dillon, C. Beck, T. Kyte, J. Kallman, and H. Rogers, "Introduction," Beginning Oracle Programming, pp. 1–8, 2002.
10. S. Sakr, M. Wylot, R. Mutharaju, D. Le Phuoc and I. Fundulaki, "Processing of RDF Stream Data", 2019.
11. T. Akidau et al., "MillWheel", Proceedings of the VLDB Endowment, vol. 6, no. 11, pp. 1033-1044, 2013. Available: 10.14778/2536222.2536229.
12. D. Abadi et al., "Aurora: a new model and architecture for data stream management", The VLDB Journal The International Journal on Very Large Data Bases, vol. 12, no. 2, pp. 120-139, 2003. Available: 10.1007/s00778-003-0095-z.
13. A. Toshniwal et al., "Storm@ twitter", Cs.brown.edu, 2019. [Online]. Available: https://cs.brown.edu/courses/csci2270/archives/2015/papers/ss-storm.pdf.
14. M. Zaharia, T. Das, H. Li, S. Shenker and I. Stoica, "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters", Dl.acm.org, 2012. [Online]. Available: https://dl.acm.org/citation.cfm?id=2342773.
15. L. Affetti, R. Tommasini, A. Margara, G. Cugola and E. Della Valle, "Defining the execution semantics of stream processing engines", Journal of Big Data, vol. 4, no. 1, 2017. Available: 10.1186/s40537-017-0072-9.
16. S. Beginning Apache Spark 2 With Resilient Distributed Datasets, "Beginning Apache Spark 2 With Resilient Distributed Datasets, Spark SQL, Structured Streaming and Spark Machine Learning library 1st ed. by Hien Luu | WHSmith", WHSmith, 2019. [Online]. Available: https://www.whsmith.co.uk/products/beginning-apache-spark-2-with-resilient-distributed-datasets-spark-sql/hien-luu/paperback/9781484235782.html.
17. APACHE. Apache Beam. 2017. Available: https://beam.apache.org/
18. Google. Google Cloud Dataflow. 2015 Available: https://cloud.google.com/dataflow/

AUTHORS PROFILE

Nirav Bhatt is working at Department of Information Technology in Chandubhai S Patel Institute of Technology, CHARUSAT. He had received degree of Master of Engineering in Computer Engineering from Dharmsinh Desai Institute of Technology and currently pursuing his Ph.D. in the area of Big Data Stream Analytics. His research interests include Database System, Data Mining and Big Data Stream Analytics. He is also a member of Computer Society of India and ACM and member of ACM Chapter at the institute. He is also coordinator of SWAYAM-NPTEL Local Chapter which is the National MOOCs portal being developed by MHRD, Govt. of India.

Dr. Amit Thakkar has received his B.E Degree in I.T. from Gujarat University in 2002 and Master Degree from Dharmsinh Desai University, Gujarat, India in 2007. He has finished his PhD in the area of Multi Relational Classification from Kadi Sarva Vishwa Vidyalaya (KSV), Gandhinagar, India in 2016. He is working as an Associate Professor in the Department of Information Technology in Faculty of Engineering and Technology, CHARUSAT University, Changa, Gujarat Since 2002. He has published more than 50 research papers in the field of Machine Learning. Data Mining. He has also published 3 books in the area of Data Mining. His current research interest includes Machine Learning, Multi Relational Data Mining, Relational Classification and Big Data Analytics. He has also severed as a member of Technical Committee at Various National and International Journals/Conferences.