A Survey of Data Stream Processing Tools

Marcin Gorawski, Anna Gorawska and Krzysztof Pasterak

Abstract In current international context boundaries set for applications are being pushed by the emergence of bursty and time-varying data streams required to be processed in near real-time. Furthermore, traditional techniques for data mining cannot be applied to data streams. Thus, stream-based applications must exhibit to excel at a plurality of requirements. According to defined rules presented in previous promulgated researches on this subject we differ stream-based applications and evaluate their aptitude to stream sources management. By this work we intend to present features and drawbacks of existing software coming from both industry and academic world, along with outlining our contribution to this field.

Keywords Data mining · Data stream processing · Real-time processing · Tool comparison

1 Introduction

A large class of applications has been distinguished in accordance to source data being generated asynchronously in an unpredictable manner. Data streams [13, 15], as they were called, are massive sequences of data that are rapid, unbounded in size, real-time and very often contain multidimensional data items, i.e., tuples. As it was outlined in [15] there are attempts to adjust classical systems as DBMS or rule engines to manage data stream processing. However, they tend to fail in terms of the most important real-time processing requirements. Consequently, applications oriented

M. Gorawski (✉) · A. Gorawska · K. Pasterak
Institute of Computer Science, Silesian University of Technology,
Akademicka 16,
44–100 Gliwice, Poland
e-mail: Marcin.Gorawski@polsl.pl

A. Gorawska
e-mail: Anna.Gorawska@polsl.pl

K. Pasterak
e-mail: Krzysztof.Pasterak@polsl.pl

T. Czachórski et al. (eds.), *Information Sciences and Systems 2014*,
DOI 10.1007/978-3-319-09465-6_31

toward processing data streams are substantially different from conventional ones, thereby ill-equipped, which cannot meet high-volume and low-latency processing criterions. Tools primarily designed to serve this purpose are far more effective and reliable in managing stream-oriented workloads.

1.1 Research Criterions

While creating data stream processing system from a scratch, following real-time processing requirements [15] ought to be addressed:

1. *In-stream processing*. Where system processes data without any additional memory operations, i.e., storing, in a non-polling processing model. While providing for 'straight-through' operations, time-intensive storage operations are omitted in processing pipeline which results in increasing system's time efficiency.
2. *Stream SQL-like language*. Built-in dedicated language where primitives and operators express continuous stream processing characteristics.
3. *Handling streams imperfections*. With handling unpredictable sources resiliency is essential in terms of performing calculation on partial, deficient, delayed, or out-of-order data.
4. *Predictable outcomes*. Processing pipeline should secure time-ordered processing of data, so when reprocessing sequence of tuples due to failure occurrence, outcome would be the same as if failover did not happen. Therefore, performing operations on stream data ought to lead to predictable, deterministic and repeatable results.
5. *Integration of current and historical data*. Although data stream processing model aims at management of current real-time data, it must also provide basis for historical data processing. By extending systems capabilities in such manner we gain possibility to identify whether data pattern observed in current online data is consistent with already established ones,
6. *Safety and availability*. Failure occurrence is a critical concern. Thus, stream-based applications have to be up and in case of failover seamlessly recover ensuring data availability and integrity.
7. *Distributed processing*. Spreading processing across multiple computing units is sufficient in terms of gaining scalability.
8. *Minimization of response time*. Even if all previous features characterize system when it is not optimized towards minimizing overheads it will not be able to react efficiently to continual inputs from various stream sources. Therefore, when escalation in number of data volumes in time is observed, system must procure responses without latency in a continuous timely fashion.

Sensorization of the world have posed many challenges gathered as a list of eight requirements. Next two sections will present applications which target those issues. In Sect. 2 five selected data stream processing systems will be shortly described, while

Sect. 3 will serve a purpose of propagating our contribution in a form of constantly maturing systems. Finally, we summarize features of described systems and provide concluding remarks.

2 Selective Overview of Stream-Based Processing Systems

2.1 Aurora and Borealis

In its early stage Aurora [4] system was built as a single site stream-based application at Brown University. Thus, it is not prepared for distributed and parallel processing. Unfortunately, it does not satisfy requirements of scalability, reliability, and data safety. Borealis processing engine [3] is a successor to the previously mentioned Aurora system. Among modules of the Borealis system there are: stream processing engine, load manager, and load shedder. Further, a mechanism of fault tolerance satisfies the need to provide safety and availability in case of system's failure, while revision processing mechanism handles tuple's imperfections by correcting erroneous ones. Apart from mentioned features it incorporates textual query definition language built on the XML standard, dynamic revision of query results, and query modification.

2.2 Apache Storm

Next system is an open-source and free product called Storm [2], which focus is set on real-time data processing with ability to distribute and parallelize computation. Moreover, the Storm cater fault tolerance in case of tuple imperfections and ensures processing even if failover occurred. With an architecture inspired by Hadoop, the Apache Storm is easy to use and what is more important it can be easily linked with most of existing queuing and database technologies. Previously mentioned systems came from academic world.

2.3 Apache Samza

Another stream-based application from Apache [1] aims at performing computation over data streams by combining Apache Kafka's messaging and Apache Hadoop NextGen MapReduce (YARN). The last ones purpose is provision of fault tolerance, processor isolation, security, and resource management. Input streams of events are decomposed and partitioned so that data flow graph is created. Each graph contains multiple streams and jobs (i.e., processing primitives), which describe different utilities of continuous queries.

2.4 Microsoft StreamInsight

Microsoft StreamInsight [5] is an extensible framework, merged with Microsoft SQL Server, which leverages continuous processing of long-running (in theory even infinite in time) streams of events. This system has been architected on the basis of CEDR [7] project. The most important aspect is ability to integrate Microsoft StreamInsight with almost any field of interest by constructing a proper processing pipeline. Logic of queries is based on a temporal time model and operator algebra. Meanwhile, the system aims at revealing patterns and trends.

2.5 StreamGlobe

StreamGlobe [14] is a grid-based P2P DSMS. With stream and result sharing mechanisms as well as early filtering and aggregation it enables to avoid redundancy in terms of data stream transmissions and computation. Moreover, by using StreamGlobe reduction of network traffic and pear load is observable.

3 Our Contribution to Data Stream Processing

In the previous section five selected systems where described shortly. In accordance to mentioned features we want to present our contribution to this field.

3.1 StreamAPAS

The first system, StreamAPAS [10, 11], implements following stream operators: selection, projection, duplicate elimination, grouping (aggregation), join, union, intersection, and difference. In addition, it contains also a stream query language, as well as a query graph optimizer.

The first important feature of the StreamAPAS system is time model—*mixed* time model [10], combining both temporal and negative [9] time models. The mixed time model was introduced in order to eliminate disadvantages of the aforementioned models.

The stream processing system StreamAPAS includes a declarative stream query language [11]. User defines format of final outcome rather than declaring subsequent steps of the algorithm (as in procedural languages). Therefore, process of defining stream query is independent of the physical architecture of the system. Moreover, the StreamAPAS language contains some features that are present in object oriented languages, which increases its functionality extension capabilities.

Another advantage is existence of attribute trees, i.e., hierarchical structures used instead of simple values for defining tuple's data, which makes organizing and managing easier.

The next important feature of the StreamAPAS system is the usage of operator partitions in a query graph optimization process. The operator's partition is also called the compound operator, which includes variety of simple (regular) stream operators. The operator partition is seen by the system as an usual operator and its internal components exchange tuples without buffering. Such objects are created when the total time or memory characteristic of certain simple operators are worse than characteristic of them bounded in one operator's partition.

3.2 THSPS

Next step in our research on the stream-based processing systems was creation of yet another system, THSPS, which supports selection, transformations (projection and mathematical operations), join, and aggregation operations. In this system our focus was set on different time model, the *tri-temporal time model* like in Microsoft StreamInsight, where each tuple time is described by three dimensions:

- *occurrence time*. Time interval when the event described by tuple has appeared,
- *validity time*. Time interval in which tuple can be analyzed by the system,
- *system time*. Timestamp representing tuple's arrival time, i.e., moment in time, when it has arrived to the system.

Among many advantages of the tri-temporal time model the main benefit is possibility of analyzing both current and historical events—which enhances views of stream sources.

Each tuple represents a real-world event, which is described by two time intervals, while the system time is used rather by the system for maintaining proper order of tuples and is not directly connected with the event itself. Therefore, operators that use multiple tuples in one cycle (i.e., join and aggregation operators) need to consider these time dimensions during their operations. When joining data is concerned, outcome tuple is characterized by both time dimensions narrowed to the common part of the corresponding dimensions of two joined tuples. Thus, the result of joining operation is an event that occurs when two input tuples overlap.

Analogical situation occurs when in processing pipeline aggregation operator is used. Input sequence of tuples after aggregation is merged into one. Outcome tuple consists of aggregated value (e.g., sum, mean value) and time intervals equal to sum of the corresponding time dimensions from all aggregated tuples. So, resulting tuple carries information about an event which lasts as long as all aggregated tuples.

Another sufficient module of the THSPS system is the query graph optimizer, which rebuilds internal structure of the graph to reduce number of unnecessarily processed tuples. This graph represents processing pipeline. For example, moving

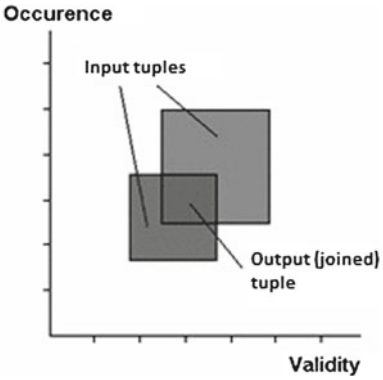

Fig. 1 Join operation in tri-temporal time model

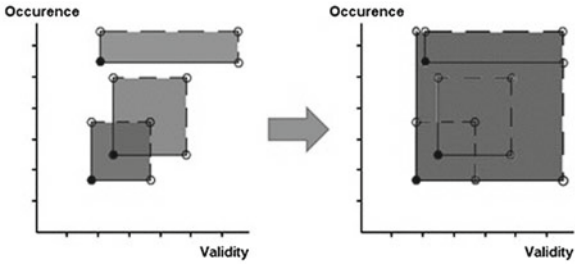

Fig. 2 Aggregation operation in tri-temporal time model

a selection before other operators does not change query's result. Moreover, it optimizes query by reducing number of tuples by eliminating those which do not satisfy selection operator's predicate. Thus, tuples will be discarded on the very beginning of their path through the graph. The major rule of the query optimizer is that operators that have the smallest *selectivity in time* (i.e., the difference between the number of input and output tuples divided by the operator life time), should be considered as candidates to transferring to the beginning of the graph.

The other feature of the THSPS system is existence of *joined tuple decorating mechanism*. Its principle is to create a tuple decorator, i.e., an object which has the same interface as a regular tuple, but holds inside two physical tuples that have been joined. Such concept was designed to reduce total number of tuples currently stored in the system, as well as to increase the speed of producing results in join operator—due to decreasing memory allocation and attribute copying operations.

3.3 AGKPStream

After collecting experiences from both previously created systems in the AGKP-Stream system [13] temporal time model was chosen as well as following stream operators: selection, projection, union, and join (in three variants: *cross-join*, *equi-join* and *theta-join*). Each operator defines two basic transformations aiming at schema or attribute modifications and reformations.

In the AGKPStream system, the temporal time model was selected. Each tuple denotes a time-lasting event that is represented by the time interval called *tuple life time* and is described by two border timestamps: beginning and end. While the tuple is valid (the current system time lays between these two timestamps) it can be processed.

All operators in the AGKPStream system form and DAG likewise in the Aurora system, each representing a single continuous query. The work of every processing primitive is managed by the scheduler, which can work in one of following modes: query or global level.

Global scheduler chooses one query each time and calls it to work, by activating its local scheduler. Query's schedulers manages operators by pointing which should work in a particular moment in time, basing on various factors.

Similarly to the THSPS system, the AGKPStream system uses joined tuple decorator mechanism. In both solutions, such mechanism enables the system to join only two tuples. However, in some special cases, e.g., when multiple join operators are connected in cascade, resulting joined tuples (when using decorating mechanism) are also formed in the same manner—subsequent decorating objects points other decorators etc.

4 Evaluation of Data Stream Processing Systems

While conducting presented research we have examined tools supportable over data stream processing paradigm. Three out of eight were created by our team during past years, while remaining five were taken as a reference point. Each system was examined in terms of requirements as follows: in-stream processing (1), stream language (2), handling stream imperfections (3), predictable outcomes (4), data safety and availability (5), integration of stored and stream data (6), distribution and scalability (7), and instantaneous outcome (8).

Table 1 presents eight requirements listed in the introduction versus their coverage in aforementioned systems, where each entry has one of five values:

- *yes*. System natively provides this capability,
- *no*. System does not support this feature,
- *possible*. This challenge was not solved in the system yet, but it is possible to do so,

- *hard*. Theoretically adaptation to this requirement is possible, but it would require substantial changes in the system,
- ? Lack of information.

Table 1 Comparison of described systems with emphasis to the eight requirements

System	Req. 1	Req. 2	Req. 3	Req. 4	Req. 5	Req. 6	Req. 7	Req. 8
Borealis	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Apache Storm	Yes	No	Yes	Yes	?	Yes	Yes	Yes
Apache Samza	Yes	No	Yes	Yes	Possible	Yes	Yes	Yes
StreamInsight	Yes	No	Yes	Yes	?	Yes	Yes	Yes
StreamGlobe	Yes	?	Possible	Yes	?	?	Yes	Yes
StreamAPAS	Yes	Yes	Yes	Yes	Hard	Yes	Yes	Yes
THSPS	Yes	Possible	Hard	Hard	Hard	Possible	No	Yes
AGKPStream	Yes	Possible	Yes	Yes	Hard	Possible	Yes	Yes

As Table 1 shows, all tested systems fulfilled condition to yield responses in a timely manner with emphasis to processing data in the 'straight-through' model. Another fundamental rules from 3 and 4 were satisfied by most of the systems, while integrating historical with real-time data and possession of native stream querying language were arised as the most problematic requirements.

5 Concluding Remarks

The aim of this paper was to supply with a selective survey of tools enabling data stream processing. Created comparative review was based on assumption that mentioned eight requirements present data stream processing model's characteristics adequately. In previous sections we have supplied tabular results of conducted research where goal was to provide user with an upfront knowledge what types of systems are available on the market and how they rise to streaming challenges.

Unfortunately, Aurora, Borealis, and Stream [6] projects are no longer active but it is not an indication of meaninglessness of research continuation. On the contrary, there are still numerous of new and subtle problems unsolved thereby our ongoing work on the topic is even more valuable. We are not only pursuing evolution of data stream processing systems, but also joining its most important objectives with other areas of our domain expertise, i.e., databases and data warehouses. With this combination first attempts to create a Stream Data Warehouse were made [12]. Moreover, many researches all over the world focus their efforts on adapting data stream processing model and big data to create new solution in presented domain, like StreamGlobe, PIPES [8], Apache Samza, and Storm, among with many other attempts.

References

1. Apache Samza official website. http://samza.incubator.apache.org/
2. Apache Storm official website. http://storm.incubator.apache.org/
3. D.J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, S.B. Zdonik. The design of the borealis stream processing engine. in: *CIDR*, pp. 277–289, 2005
4. D.J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik, Aurora: a new model and architecture for data stream management. VLDB J. **12**(2), 120–139 (2003)
5. M. Ali, B. Chandramouli, J. Goldstein, and R. Schindlauer. The extensibility framework in microsoft streaminsight. in: *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE '11, IEEE Computer Society*, pp. 1242–1253. 2011
6. A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, J. Widom, Stream: the stanford stream data manager. Technical Report 2003–21, Stanford InfoLab, 2003
7. R.S. Barga, J. Goldstein, M.H. Ali, M. Hong. Consistent streaming through time: a vision for event stream processing. in: *CIDR*, pp. 363–374. www.cidrdb.org
8. M. Cammert, C. Heinz, J. Krmer, A. Markowetz, B. Seeger, Pipes: a multi-threaded publish-subscribe architecture for continuous queries over streaming data sources. Technical report, 2003
9. T.M. Ghanem, M.A. Hammad, M.F. Mokbel, W.G. Aref, A.K. Elmagarmid, Incremental evaluation of sliding-window queries over data streams. IEEE Trans. Knowl. Data Eng. **19**(1), 57–72 (2007)
10. M. Gorawski, A. Chrószcz, Query processing using negative and temporal tuples in stream query engines. in: *Advances in Software Engineering Techniques—4th IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2009, Revised Selected Papers, volume 7054 of Lecture Notes in Computer Science*, (Springer, 2009) pp. 70–83
11. M. Gorawski, A. Chrószcz, StreamAPAS: query language and data model. in: *2009 International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2009, IEEE Computer Society*, pp. 75–82, 2009
12. M. Gorawski, A. Gorawska, Research on the Stream ETL Process. in: *Beyond Databases, Architectures, and Structures, volume 424 of Communications in Computer and Information Science*, (Springer, 2014) pp. 61–71
13. M. Gorawski, A. Gorawska, K. Pasterak, Evaluation and development perspectives of stream data processing systems, in: *Computer Networks*, vol. 370 (Springer, Berlin, 2013), pp. 300–311
14. R. Kuntschke, B. Stegmaier, A. Kemper, A. Reiser, Streamglobe: processing and sharing data streams in grid-based p2p infrastructures. in: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB*, pp. 1259–1262. ACM, 2005
15. M. Stonebraker, U. Çetintemel, S. Zdonik, The 8 requirements of real-time stream processing. SIGMOD Rec. **34**(4), 42–47 (2005)