Journal of Theoretical and Applied Information Technology
15th May 2021. Vol.99. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CURRENT ISSUES AND METHODS OF EVENT PROCESSING IN SYSTEMS WITH EVENT-DRIVEN ARCHITECTURE

[1]VALERII VLADIMIROVICH PETROV, [2]ANNA VICTOROVNA GENNADINIK, [1,3]ELENA YUREVNA AVKSENTIEVA, [1]KONSTANTIN VLADIMIROVICH BRYUKHANOV

[1]Faculty of Software Engineering and Computer Engineering, ITMO University, Kronverksky Pr., 49, Saint-Petersburg, 197101, Russia

[2]Department of Computer Science, Hamburg University of Applied Sciences, Berliner Tor, 5, Hamburg, 20099, Germany

[3]Information Technology and E-Learning Department, The Herzen State Pedagogical University of Russia, Moika Emb., 48, Saint-Petersburg, 191186, Russia

E-mail: mu_valera@mail.ru

ABSTRACT

Systems and methods of event processing are used in many large-scale research projects of European Organization for Nuclear Research, major IT and financial corporations, IoT and many other companies for development of event-driven architectures. Scenarios of application of methods and systems of event processing are numerous, they become more and more popular every year, but part of the mentioned systems is limited due to common event processing issues. Therefore, the issues occurring upon event processing have been studied. This work analyzes the notions of event processing, event processing methods, history of the subject field, urgent issues of event processing methods. In addition, the approaches to their solution are proposed, differences in event processing methods are exemplified, drawbacks of the methods are highlighted. This work analyzes the following problems, which can occur while designing event processing system: processing of out-of-sequence events; occurrence of duplicates; collision upon event processing; distributed fault-tolerant architecture; multi-threaded event processing; adaptive circuits of load balancing; monitoring of event processing application. Each aforementioned problem is briefly described. Several compromise solutions are discussed and tested with the usage of the test bench; it's aimed at smoothing of consequences of the application of the existing approaches. Some of the methods could lead to the performance degradation. For research purposes were initiated Fuzzy testing for tracking the potential failures and recoveries.

Keywords: *Event Processing, Complex Event Processing, Stream Event Processing, Event Tracing*

1. INTRODUCTION

Event processing is a method of tracking and processing streams of data about events. This work considers urgent issues of event processing in systems with event-driven architecture and their optimum solutions. Event processing is applied in many fields, such as Monitoring, IoT, Security, Payment Processing, Big Data, and others. Event processing is applied in the fields, which require online response to input data flow and data processing with minimum delays.

Event processing becomes more and more popular due to data amount increasing in geometric sequence. These data should be processed, analyzed, and certain regularities should be determined. This subject is closely connected with the methods and techniques of machine learning as well as with the approaches to data analysis. In addition, processing and response to events are used in design of event-driven architecture of software.

Depending on the applied scenarios, various problems can occur upon design and operation of event processing systems. Respectively, various approaches or their combination should be used to balance these problems.

Prior to review the subject field, it would be appropriate to define the notions of event and event processing.

In general case, **an event** is everything that occurs or is considered as occurring, for instance, financial transaction or aircraft landing [1].

In informatics, **an event** (object of event, message, array) is the object being the record for subsequent processing by system [2], for instance, payment entry, E-mail confirmation of aircraft landing.

Event processing is the procedure, which analyzes various methods of data processing concerning events.

This procedure is subdivided into four methods of data processing: Complex Event Processing; Stream Event Processing; Cognitive Event Processing; Hybrid Event Processing.

This work is an integrated study, since it includes analysis of the subject area in the first section and analysis of problems occurring upon design and operation of event processing systems in the second section.

The result of the study is the determination of problems occurring upon design and operation of systems with event-driven architecture and the most optimum methods of their solution.

2. LITERATURE REVIEW

The issues of event processing are important in modern information society with developed IT infrastructure, which generates huge data arrays. D. Luckham should be mentioned, since he pioneered the field of event processing. His works [1-3] established initial aspects for development of event processing systems and event-driven architecture.

All the related works mentioned in this section were selected for study by the following criteria:

- the main research interest in work should be an issue in event processing;
- this issue should be connected to this study;
- work should include at least 3 variational approaches for challenging the issue.

Foreign researchers concentrate their attention on the issues of development of highly available architecture [4-6], distributed architecture [7-9].

Proletarskii A.V., Berezkin D.V., Gapanyuk Yu.E., Kozlov I.A., Popov A.Yu., Samarev R.S., Terekhov V.I. paid attention to the methods of situational analysis and visualization of data flows [10, 11], J. Lang and Z. Capík analyzed forecasting on the basis of hybrid methods upon processing of complex events [12, 13].

The issues of data balancing in event processing systems were discussed by N. Zacheilas, N. Zygouras, N. Panagiotou, V. Kalogeraki, and D. Gunopulos, M. A. U. Nasir, G. F. Morales, D. Garcia-Soriano, N. Kourtellis, M. Serafini: balancing of high data amount upon distributed stream processing [14], load balancing for mechanisms of distributed stream data processing [15].

Monitoring of event processing systems was discussed by M. R.N. Mendes, P. Bizarro, P. Marques: studying performance of event processing systems [16], measuring of performance of systems of complex event processing [17, 18].

Current issues in CEP were also discussed by I. Flourisa et al. [19], ordering in event processing systems was discussed by M. Li et al. [20], parallel complex event processing was discussed by M. Hirzel [21].

The hypothesis of this research is that a combination of the event processing approaches may lead not only to the increased performance of the system but to the degradation of the whole it and to increase the time to recovery it. Worth noting that most of the problems with the event processing system can be avoided at the phase the pre-architectural design.

An event processing system with event-driven architecture was considered as the object of research, methods of event processing were considered as the subject of research to provide high availability, load balancing, monitoring, correct event processing, event processing with minimum delays.

3. MATERIALS AND METHODS

This work was based on practical experience, multi-year research, and registered patents in the following fields: Complex event processing; Stream event processing [22, 23]; Cognitive event processing [24-27].

The mentioned above patterns were chosen as a reference because it contains:

- Complex Event Processing;
- Multi-Worker Processing;
- Integrations with external systems.

These criteria are extremely important, because the modern event processing system consists at least 2 mentioned criteria.

It was proved on practical and scientific level in [3, 5, 25, 28], that application of event processing system could solve various problems of big data processing with minimum delays, i.e. in online mode.

A portion of the considered methods was verified using test bench. Notifications from various monitoring systems were used as event with their subsequent processing by system. The overall

processing pipeline have included the 4 following layers:

- Normalization;
- Enrichment;
- Correlation;
- Presentation.

For test purposes, all events have been tagged with the appropriate worker name and timestamp to track the event processing performance. Also, were performed a Fuzzy testing for the all pipeline to imitate the potential failure and recovery.

4. RESULTS

This work is aimed at the determination of problems occurring upon the design and operation of systems with event-driven architecture and the methods of their solution.

Approaches to complex event processing were developed in Stanford University from 1989 to 1995 in order to analyze event-driven simulation of distributed systems.

These investigations started from development of new language Rapide, which was intended for simulation of events in distributed systems controlled by events.

Complex event processing (CEP) is the method of tracing and processing of data stream from numerous sources of events in order to identify the most significant events or their combination and making decisions in real time [2].

At the same time, the main **features** of **CEP** are highlighted: events in distributed systems can occur independently on each other; they can occur both simultaneously and at different time; occurrence of one event can result in occurrence of another event.

Event stream processing (ESP) is the processing of events arranged in time and continuously supplemented by new data [2].

The ESP can be exemplified as follows: payment processing; detection of fraudulent activity; detection of abnormalities; metrics processing.

ESP is usually applied, when the application scenario requires for processing of data points distributed along the time scale. The order and distribution of data along the time scale allow to analyze trends, to search for similar and repeated events, thus enabling highlighting of a data stratum important for final user.

Both approaches, CEP and ESP, are required for efficient solution to different problems.

Their difference is as follows: if it is required to analyze data stream online, then ESP is more suitable for such problem. However, if it is required to process event cloud, then CEP is more suitable [29].

The main difference of event flow from cloud is that the event flow is arranged in time (for instance, quotation of securities market). In the cloud, events are generated due to operation of numerous systems in various components of information systems. The event cloud can contain several event streams. Event stream is a peculiar type of cloud.

Cognitive event processing is the method of event processing, which uses cognitive approaches to expat potentials of CEP systems, this method is based on cognitive calculations [30].

Cognitive calculations are comprised of forecasts of certain events or event groups using methods and models of machine learning [30].

Machine learning models which are used for cognitive event processing: Decision tree; Bayesian classification; Neural networks; Genetic programming; Support vector machines; Symbolic regression.

Decision tree is one of the basic techniques. It contains simple rules, which can be expressed in natural language. This method is the most suitable for forecasts by repeated data.

Neural networks are one of the methods characterized by universal functions of approximation due to capability to compare input and output data. They can be subdivided into various methods of learning, network topology, etc. Successful scenarios of neural networks imply forecasts not of precise values but of definite differential vector (for instance, price increase or drop).

Genetic programming involves attempt to present operation of genetic algorithms during evolution. This is the process of population selection, when more adapted generations have more chances for reproduction. In this case, the fitness function is used for population improvement.

Support vector machine is a field of machine learning, which is the most suitable for classification as well as for forecast of linear and nonlinear data.

Hybrid event processing (HEP) is the method of event processing, combining peculiarities of other methods of event processing [13]. Such method of event processing is intended for complex geo-distributed systems with adaptive load balancing; it is applied when certain scenarios cannot be implemented using one of the methods of event processing. As a rule, development of event processing system starts from definition of the event model structure.

Event model structure

A single established definition of event model structure is unavailable, most authors interpret this term in different ways. In [6] the structure is described as a tuple **e=<s,t>**, where **e** is the considered event, **s** is its attributes, **t** is the list of time marks.

When the event model structure is defined, it is required to determine how the events will be selected from event stream/cloud and what model will be used for detection of events.

At this step, the model could face the following problems: event order in the stream was violated; duplicates exist in the event stream.

There are several methods to solve the problem with out of order events:

Creation of special buffer: buffer stores all input events until definite condition preset by user is satisfied. Then the buffer content is sorted in terms of definite key and redirected for further processing. The main drawback of this approach is that when the number of events increases, the buffer increases accordingly, thus, the delays of event processing will also increase. The K-slack algorithm can also be applied. The key concept of this method is that event processing can be delayed by maximum of K time units with subsequent sorting by time mark. This method can be efficiently applied in the case of insignificant network delays. In the case of significant delays, the buffer will be deallocated also with delays.

Another approach is development of logics, when such events could be neglected. As a rule, such approach is very limited in terms of scenarios of its use, moreover, it could significantly complicate the other logics of operation of event processing system and lead to increase in expenses for maintenance of such system as well as to increase in the number of errors due to general complexity of the system.

The problem of existence of duplicates in the event stream can be solved using the following methods of event processing:

The simplest possible method is ID generation by several fields [31, 32], including timestamp of the event. Herewith, the deduplication would require for presence of all events in one place. Therefore, it would be reasonable to direct and to filter event stream from one source which generates the events in order to facilitate operation of the deduplication and to provide opportunity for further distributed architecture design. The main drawback of this method is that in the case of large stream of input events, the buffer could be overfilled. All depends on scenarios of use, amount of input events, and possibility of existence of duplicates in the system.

As a rule, the following thresholds are set for buffer cleaning from events: in terms of number of events in the buffer; of time; of ratio of occupied to free memory.

During operation of event processing system, it was revealed that collisions could occur. Collisions upon event processing can occur in the case of: data replication; use of multi-threaded processing.

Collisions upon data replication can be caused by different reasons: asynchronous replication; delay in data delivery.

Collisions upon asynchronous replication can occur in the case of simultaneous access to one and the same object at one and the same time. Such collision can be resolved by blocking or optimistic concurrency. In the case of blocking, the object is blocked until termination of the previous operation. In the case of optimistic concurrency, the record is read with fixation of number of object version and subsequent verification is performed upon recording: whether the object version has changed. Delays in data delivery could occur upon network unavailability, loss of packages, failure of packages. They can be avoided by decreasing the distance between objects as well as by minimization of number of network devices connecting server group. In the case of multi-threaded event processing, collisions are possible when different streams process and update simultaneously the event, which was stored in the state of event processing system. This problem can be smoothed by means of the following approach: all events from one essence can be processed only in the frames of one stream. The disadvantage of this approach is as follows: in the case of high stream of events, the approach will be inefficient due to artificially limited parallelism (Figure 1).

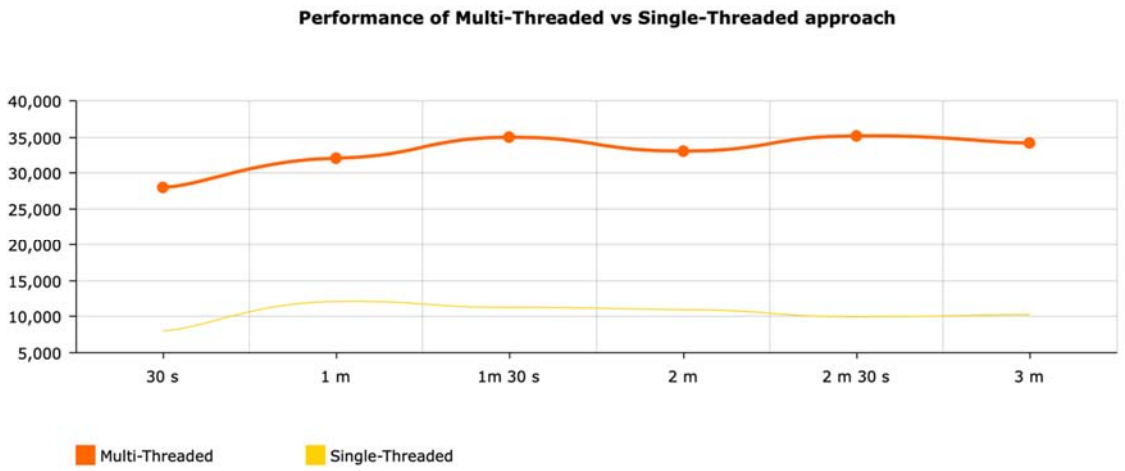

Figure 1: Performance of Multi-Threaded vs Single-Threaded Approach

Distributed and fault tolerant architecture of event processing system can be developed using the following main and general **recovery methods after failure**: Precise Recovery; Rollback recovery; Gap recovery.

Precise recovery nearly ideally hides consequences of fault, including possible delays upon data processing. This method is suitable for applications, which require the same event stream as before the fault. That is, complete identity and idempotence are guaranteed.

Rollback recovery: in the case of this recovery method, duplicates can occur, the initial order of events can be changed, as well occurrence of delays upon event processing. For complete recovery, the state of recovery is used, which can cause the aforementioned problems.

Gap recovery is one of the most unreliable recovery methods intended for applications which operate exclusively with the latest data; in this case, removal of old data is acceptable for reduced time of recovery and execution of program code.

High accessibility of event processing system is provided by the following methods: Active/standby architecture of workers (Service Worker); Passive/standby architecture of workers; Hybrid architecture.

In Active/standby architecture, two or more copies of the node responsible for data processing operate independently on each other at different actual servers. When one of the nodes fails, the other will not be affected, data replication between the two nodes takes place by multi-plexing, which in its turn can invoke occurrence of duplicates.

In Passive/standby architecture, the main node periodically copies state (state is a set of events required for further data processing and it is available locally for each worker in the system) to passive node by means of messages of reference point, and the recovery takes place on the failed node.

Upon state copying, the following problems can occur: further forwarding of data from queue; uncertainty concerning the data to be included in messages of reference point; non-unique initiation of reference point messages.

These problems are classic upon synchronization by means of reference points.

In addition, when Active/standby and Passive/standby approaches are used for high accessibility, the following problems can appear: unavailability of a worker; overload of a worker.

These problems occur as a result of: network issues; excessive use of CPU/IO; wavelike peak loads.

Partially these problems can be eliminated by adaptive load balancing.

As a rule, most systems of event processing support at least one of the aforementioned methods.

The advantage of Active/standby architecture in comparison with Passive/standby architecture is that the system is recovered faster, however, additional load upon replication is introduced (Figure 2).

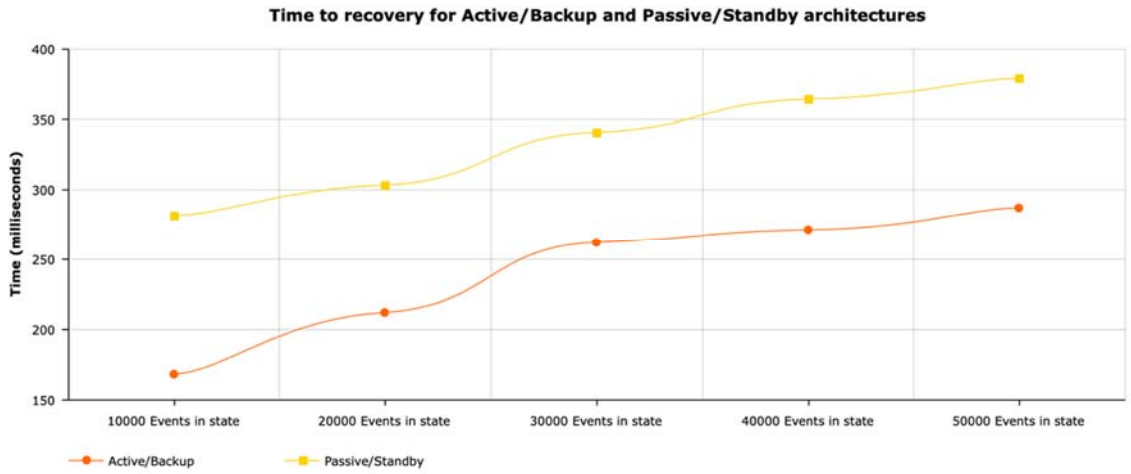

Figure 2: Time to Recovery for Active/Backup and Passive/Standby Architectures

Adaptive circuits of load balancing

The event processing systems are characterized by the problem of complete and homogeneous distribution of load across all nodes responsible for event processing. First of all, it should be determined, for which system of event processing the load balancing will be used. The systems can be subdivided into two types: with state and without state.

The problem of adaptive balancing can be smoothed using mechanisms on provision of dynamic distribution of load. This problem is related to NP complex problems; in particular, this is a classic problem of scheduling theory.

There are definite amounts of similar units, which can process events, and it is required to uniformly distribute the load across these units in order to minimize certain performance metrics.

In particular, adaptive circuits of load balancing can be subdivided into the following methods: Load balancing on the basis of node capacity; Load balancing on the basis of data stream.

A drawback of the adaptive dynamic balancing of load is that under such conditions, it is extremely difficult to recover event stream if it was confused. Moreover, it is extremely difficult to avoid possible duplicates generated by one and the same object because there are no guarantees that the next message will be delivered to the same node, which processed the initial message and not the duplicate. In addition, upon existence of messages of different size in the stream, the adaptive load balancing will be complicated because tracing of all nodes, which process events in distributed system, is a very difficult task.

Load in event processing system can be balanced by round-robin mechanism. During operation of round-robin mechanism, a new message is forwarded to each next processor, this is performed in cycle. This mechanism is highly suitable to balance load in event processing system without state. Another important problem in adaptive load balancing is the problem of nonuniformly distributed events from the object in total data set. This problem was smoothed by means of algorithm of double selection (Figure 3) [14].

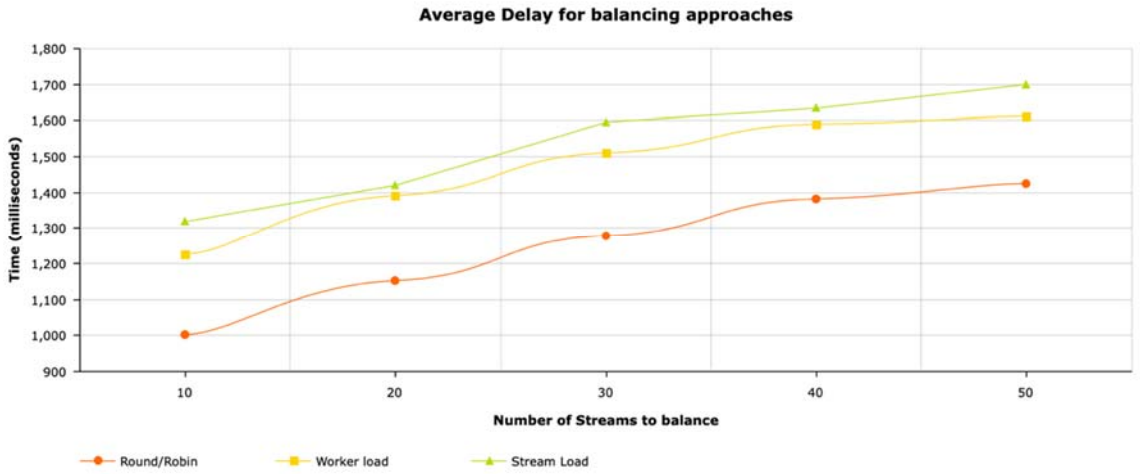

Figure 3: Average Delay for Balancing Approaches

Monitoring of event processing application is comprised of three main methods of monitoring: Sampling and processing of metrics of event processing system; Active validations; Tracing.

In order to trace the performances of event processing applications, the following metrics are usually acquired:

- total capacity (number of processed events per unit time);
- partial capacity (per each rule/set/group of rules);
- memory consumption by operations working with buffer;
- delays in event processing;
- synchronization of state (upon highly available architecture).

The aforementioned metrics are collected by monitoring agent or forwarded directly to monitoring system: for their subsequent processing and triggering in the case of problems in event processing system. Active validations are required to measure average delays, which can occur upon event processing, as well as for passing of all pipeline of the event processing system. This is especially important at high number of integrations with all possible event sources. As a rule, these validations are initiated by certain interval or threshold. If one of validations fails, then, prior to notification about the problem to user, it would be reasonable to be sure that this is not a network or another failure. This can be verified by sending similar validation several times from the same system.

In addition, active validations can assist upon detection of faults of certain nodes or a part of event processing system. In general case, the active validations should be governed by most rules for correct tracing of the system indicators.

A problem can occur upon operation of active validations in event processing system: in the case when scenarios of applications do not assume occurrence of additional delays upon event processing. Any active validation introduces minor additional load and delay upon event processing, which can be critical. In this case, it is possible to decrease the number of validations, the number of pipelines which should be traced, the number of rules passed by an event generated by active validation.

Another method of monitoring is tracing. Tracing is one of the most recent and popular monitoring methods of distributed applications. It is required for visual presentation of sequence and time of transaction, with which service and at which step the relation was generated. Using the tracing services, it is possible to plot the map of interrelations of all services, which is especially useful in the case of failures of certain nodes. Distributed tracing is universal and can be applied to trace transactions in various programming environments and languages. This universality is applied for decompression of application into the operation environment.

Two main types of tracing are available: Whitebox; Blackbox.

In addition, two main approaches are available for organization of tracing system: Agents, Libraries.

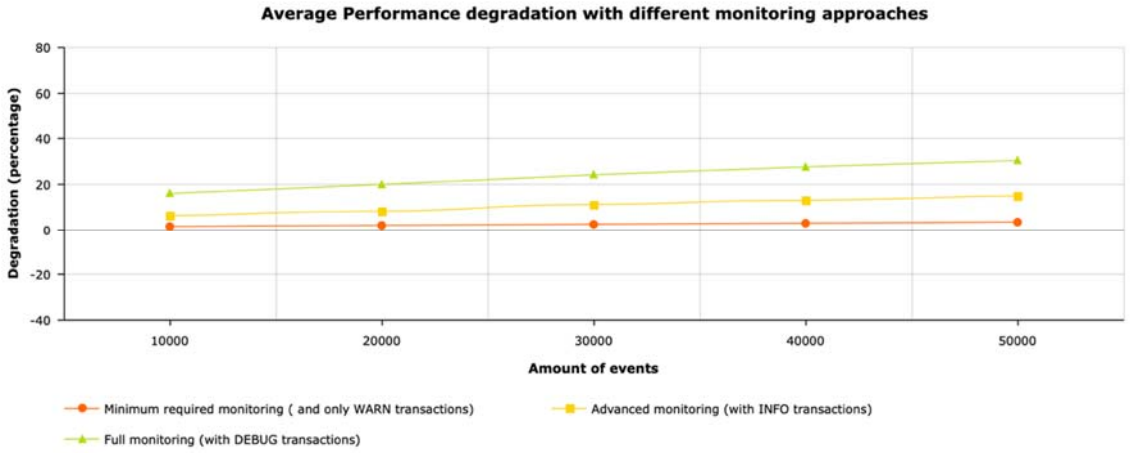

Figure 4: Average Performance Degradation with Different Monitoring Approaches

Both these approaches are used in various scenarios. The approach based on library assumes that the developer would write instrument code using API library for context propagating, which will be used for tracing of transactions. Exactly this type of tracing supports the combined use of numerous programming languages and heterogeneous architecture of application.

The agents are used based on external process or processes which instrument code during execution. There are two types of agents for tracing:

- code implementation into service to create trace using certain set of functions;
- implementation of in-process agent, which is imported into the execution environment for tracing of certain user defined actions of the system.

The difficulties, which can occur upon the use of tracing tools, are as follows:

- high amount of integrations for tracing should be implemented into original code and application architecture at initial step;
- possible decrease in performance by several per cent due to the use of tracing tools;
- maintenance, updating, and supplemental implementation of mechanisms for tracing organization into new system functionality (Figure 4).

5. DISCUSSION

Numerous difficulties exist upon design and operation of event processing systems. They can be classified as follows:

- those related with application architecture;
- those related with peculiarities of event processing;
- those related with application performance.

Nearly each solution to the problems of event processing is compromise. Thus, it is required to consider for a scenario of use of such system, available resources and limitations, which could occur upon application of methods hiding such difficulties. This study is aimed at general discussion of these problems without being bound to specific scenarios of use. Furthermore, in this study were tested few compromise options. Such as:

- deduplication (by generated ID);
- APM (tracing);
- Multi-threaded event processing with state (optimistic concurrency).

On the other hand, a few monitoring approaches were tested. The results you could find in Figure 4. All compromise solutions that were mentioned in this research were also approbation in practice.

A part of the selected methods for designing the event processing system also could lead to degradation of the performance, see Fig. 1, Fig. 2, Fig. 3. The limitation of this research based on the data which has been used in event processing system. It's a monitoring data from the application, hosts, network devices, e.g. So, it means that a part of the mentioned compromise options may be applicable for the event processing applied to the monitoring.

Were proposed a new method to process the data, it has a compromise usage, the approach might be inefficient due to artificially limited parallelism.

This study could be compared with the following research [6]. It was conducted to analyze the platforms and approaches to process the events. This

research is rich with details of query processing, load balancing algorithms (without adaptive schemas, which were researched in current paper), distributed monitoring capabilities, elastic query planning in cloud platforms. In this research was also discussed parallel CEP without multi-threaded event processing. In addition, a few deduplication schemas could be found in [33], worth to notice, that there is not discussed the presence or absence of the state (which is highly required for some of deduplication strategies). Some of mentioned approaches were also used in the following patents – [24-27]. Mostly, they are about designing appropriate event processing systems.

The paper about the parallelization and elasticity in event processing [5], has a great overview of the most mechanisms that might be used in stream processing platforms to parallel your computations with considering the usage of state. Furthermore, there was also discussed the data computation on the different workers/subset of workers (grid, cluster, cloud, etc.). Somehow it neglects the optimistic concurrency approach which is a key thing in high-performance computation in event processing system with the state.

Current research has a lack of the Artificial Intelligence usage, which is partially covered by [12, 29, 33-39]. In the current research the working models of these methods were discussed.

AI enables to process a bulk of data in real-time. Through this, AI provides meaningful insights that can solve recurring issues in systems with event-driven architecture. Businesses can gain a great number of benefits by using artificial intelligence. For instance, businesses can identify inconsistencies in their operations and anomalies in their patterns to re-strategize their processes. Not just this, but through the in-depth analysis provided by artificial intelligence, businesses can also determine the root cause of problems that they are facing. Using explorative and predictive data analysis, businesses can minimize risks and maximize the effectiveness of their business decision-making process. With this, businesses can not only capitalize on short-term opportunities but also boost profits and revenues in the long-run.

This study can be used for analysis of event processing in systems with event-driven architecture aiming at development of new efficient methods. It is comprised of review of subject area; discussion of important problems with approaches to their solution; scenarios of use; limitations, which could occur upon event processing; methods of monitoring of operation of event processing system. It also includes the approbation of compromise options.

It should be mentioned that the work analyzes classical problems occurring more often upon design and operation of such systems.

6. CONCLUSION

On the basis of the study, it is possible to conclude that the important issues of **Event processing** in systems with event-driven architecture which should be solved are as follows:

- processing of out-of-sequence events;
- occurrence of duplicates;
- collisions upon event processing;
- distributed fault-tolerant architecture;
- multi-threaded event processing;
- adaptive circuits of load balancing;
- monitoring of event processing application.

Trend for use of systems and methods of event processing is being increased every year due to increase in data amount generated by various applications, sensors, IoT devices, systems, etc. Urgent problems of the methods of event processing were revealed due to this study.

Each method was briefly described, several compromise solutions were considered and tested (see above, plots of bench testing). The hypothesis of the research has been conducted. Prior to initiation of development and operation of the event processing system, it is necessary to define scenarios of application, the essence of the event, what type of processing is required, what type of highly available architecture and monitoring will be used for system operation.

In addition, it would be reasonable to preset certain limitations for the system, because depending on the selected method, certain compromises should be applied related with solution to any problems upon event processing.

A part of the selected methods also could lead to degradation of the performance, see Fig. 1, Fig. 2, Fig. 3.

The considered problems are important because up till now there is no complete and steady solution to the problems considered in this study.

Further study could be devoted to more detailed consideration of certain problems exemplified by visual verification using actual data.

Important to consider, that the use of artificial intelligence (AI) methods in event processing is seen as a very promising direction for future research.

REFRENCES:

[1] D. Luckham, "Event Processing for Business: Organizing the Real-Time Enterprise", John Wiley & Sons, Hoboken, New Jersey, 2011.

[2] D. Luckham, "The power of events: An introduction to complex event processing in distributed enterprise systems", Springer, Berlin, Heidelberg, 2008, doi: 10.1007/978-3-540-88808-6-2.

[3] D. Luckham, "The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems", Addison-Wesley Professional, Boston, MA, 2002.

[4] Z. Zhang, Y. Gu, F. Ye, H. Yang, M. Kim, H. Lei, and Z. Liu, "A Hybrid Approach to High Availability in Stream Processing Systems", *2010 IEEE 30th International Conference on Distributed Computing Systems*, 2010, pp. 138-148, doi: 10.1109/ICDCS.2010.81.

[5] H. Röger, and R. Mayer, "A Comprehensive Survey on Parallelization and Elasticity in Stream Processing", *ACM Comput. Surv.*, Vol. 52, No. 2, 2019, pp. 1-37, doi: 10.1145/3303849.

[6] N.P. Schultz-Møller, M. Migliavacca, and P. Pietzuch, "Distributed complex event processing with query rewriting", *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems - DEBS '09*, 2009, p. 1, doi: 10.1145/1619258.1619264.

[7] J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik, "High-Availability Algorithms for Distributed Stream Processing", *21st International Conference on Data Engineering (ICDE'05)*, 2005, pp. 779-790, doi: 10.1109/ICDE.2005.72.

[8] S. Jayasekara, S. Kannangara, T. Dahanayakage, I. Ranawaka, S. Perera, and V. Nanayakkara, "Wihidum: Distributed complex event processing", *J. Parallel Distrib. Comput.*, Vol. 79-80, 2015, pp. 42-51, doi: 10.1016/j.jpdc.2015.03.002.

[9] G. Soundararajan, K. Voruganti, L. Bairavasundaram, and V. Mathur, "Distributed event processing method and architecture", US8898289B1, 2014.

[10] P. Figueiras, H. Antunes, G. Guerreiro, R. Costa, and R. Jardim-Gonçalves, "Visualisation and Detection of Road Traffic Events Using Complex Event Processing", *Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition. Volume 2: Advanced Manufacturing*, Pittsburgh, Pennsylvania, USA. November 9-15, 2018, doi: 10.1115/IMECE2018-87909.

[11] A.V. Proletarskii, D.V. Berezkin, Yu.E. Gapanyuk, I.A. Kozlov, A.Yu. Popov, R.S. Samarev, and V.I. Terekhov, "Methods of Situation Analysis and Graphical Visualisation of Big Data Streams," *Her. Bauman Moscow State Tech. Univ. Ser. Instrum. Eng.*, Vol. 2, No. 119, 2018, doi: 10.18698/0236-3933-2018-2-98-103.

[12] J. Lang, and Z. Capik, "Prediction based on hybrid method in complex event processing", *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2014, pp. 315-320, doi: 10.1109/SAMI.2014.6822430.

[13] Y. Ozawa, T. Koyanagi, M. Abe, and L. Zeng, "A Hybrid Event-Processing Architecture based on the Model-driven Approach for High Performance Monitoring", *2007 IEEE Congress on Services (Services 2007)*, 2007, pp. 324-331, doi: 10.1109/SERVICES.2007.6.

[14] L. Chen, O. Villa, S. Krishnamoorthy, and G. R. Gao, "Dynamic load balancing on single- and multi-GPU systems", *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, 2010, pp. 1-12, doi: 10.1109/IPDPS.2010.5470413.

[15] M.A.U. Nasir, G. De Francisci Morales, D. Garcia-Soriano, N. Kourtellis, and M. Serafini, "The power of both choices: Practical load balancing for distributed stream processing engines", *2015 IEEE 31st International Conference on Data Engineering*, 2015, pp. 137-148, doi: 10.1109/ICDE.2015.7113279.

[16] M.R.N. Mendes, P. Bizarro, and P. Marques, "A Performance Study of Event Processing Systems", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, pp. 221-236.

[17] T. Grabs, and M. Lu, "Measuring Performance of Complex Event Processing Systems", *The Third TPC Technology conference on Topics in Performance Evaluation, Measurement and Characterization*, 2011, pp. 83-96, doi: 10.1007/978-3-642-32627-1_6.

[18] M.R.N. Mendes, P. Bizarro, and P. Marques, "Benchmarking event processing systems", *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering - WOSP/SIPEW '10*, 2010, p. 259, doi: 10.1145/1712605.1712652.

[19] I. Flourisa, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, and M. Mock, "Issues in Complex Event Processing: Status and Prospects in the Big Data Era", *Journal of Systems and Software*, Vol. 127, 2016, pp. 217-236.

[20] M. Li, M. Liu, L. Ding, E.A. Rundensteiner, and M. Mani, "Event Stream Processing with Out-of-Order Data Arrival", *Proc. 27th Int. Conf. on Distributed Computing Systems Workshops*, 2007, pp. 67-74.

[21] M. Hirzel, "Partition and compose: parallel complex event processing", *Proc. 6th ACM Int. Conf. on Distributed Event-Based Systems (DEBS '12)*, 2012.

[22] R.S. Samarev, "Survey of streaming processing field", *Trudy ISP RAN/Proc. ISP RAS*, Vol. 29, No. 1, 2017.

[23] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: distributed stream computing platform", *Proc. IEEE Int. Conf. on Data Mining Workshops*, 2010.

[24] S. Nielsen, C. Curtis, and F. Jeffrey, "Systems and methods for complex event processing of vehicle information and image information relating to a vehicle", U.S. Patent No. 8,560,164.

[25] S. Nielsen, C. Curtis, and F. Jeffrey, "Systems and methods for complex event processing based on a hierarchical arrangement of complex event processing engines", U.S. Patent No. 8,463,487, 2013.

[26] A.C. Biazetti, A.B. Darney, E.J. Dobner, M. Feridun, K.L. Gajda, T. Gschwind, M. Moser, B.D. Pate, and M.E. Phelps, "Processing multiple heterogeneous event types in a complex event processing engine," U.S. Patent No. 8,589,949, 2013.

[27] P.-C. Chen, J. Huang, and C.-H. Hsu, "Method and system for complex event processing", U.S. Patent No. 7,457,728, 2008.

[28] A. Hartanto, F. Farikhin, and S. Suryono, "Real-time vehicles velocity monitoring and crossroads evaluation using rule-based RESTful maps API service", *J. Phys. Conf. Ser.*, Vol. 1524, 2020, p. 012016, doi: 10.1088/1742-6596/1524/1/012016.

[29] D. Luckham, "What's the Difference Between ESP and CEP?", Real Time Intelligence & Complex Event Processing, 2020, https://complexevents.com/2020/06/15/whats-the-difference-between-esp-and-cep-2/

[30] J. Yang, M. Ma, P. Wang, and L. Liu, "From Complex Event Processing to Cognitive Event Processing: Approaches, Challenges, and Opportunities", *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015, pp. 1432-1438, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.258.

[31] Snowplow Team, "Dealing with duplicate event IDs", Snowplow, 2015, https://snowplowanalytics.com/blog/2015/08/19/dealing-with-duplicate-event-ids/

[32] K. Gunia, "Event Sourcing Projections Patterns: Deduplication Strategies", Domain Centric, 2019, https://domaincentric.net/blog/event-sourcing-projection-patterns-deduplication-strategies

[33] A. Artikis, G. Paliouras, F. Portet, and A. Skarlatidis, "Logic-Based Representation, Reasoning and Machine Learning for Event Recognition", *DEBS '10: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, July 12-15, 2010, Cambridge, UK, doi: 10.1145/1827418.1827471

[34] A.Y. Sun, Z. Zhong, H. Jeong, and Q. Yang, "Building complex event processing capability for intelligent environmental monitoring", Environmental Modelling & Software, Vol. 116, 2019, pp. 1-6, doi: 10.1016/j.envsoft.2019.02.015.

[35] C. Mayer, R. Mayer, and M. Abdo, "StreamLearner: Distributed Incremental Machine Learning on Event Streams: Grand Challenge", *DEBS '17: Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, June 2017, pp. 298-303, doi: 10.1145/3093742.3095103.

[36] N. Mehdiyev, J. Krumeich, D. Enke, D. Werth, and P. Loos, "Determination of Rule Patterns in Complex Event Processing Using Machine Learning Techniques", *Procedia Computer Science*, Vol. 61, 2015, pp. 395-401, doi: 10.1016/j.procs.2015.09.168.

[37] C. Dousson, and P. Le Maigat, "Chronicle Recognition Improvement Using Temporal Focusing and Hierarchization", *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, January 6-12, 2007, pp. 324-

329.

[38] C. Cabanillas, A. Curik, C. Di Ciccio, M. Gutjahr, J. Mendling, J. Prescher, and J. Simecka, "Combining Event Processing and Support Vector Machines for Automated Flight Diversion Predictions", *Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management*, Vienna, Austria, 2014, pp. 45-49.

[39] L. J. Fülöp, G. Tóth, La. Vidács, Á. Beszédes, H. Demeter, and L. Farkas, "Predictive Complex Event Processing: a Conceptual Framework for Combining Complex Event Processing and Predictive Analytics", *BCI '12: Proceedings of the Fifth Balkan Conference in Informatics*, September 2012, pp. 26-31, doi: 10.1145/2371316.2371323.

[40] P. Vincent, "Event processing at the large hadron collider", TIBCO Software Inc., 2011, https://www.tibco.com/blog/2011/11/20/event-processing-at-the-large-hadron-collider/