UDC 004.83
Original Scientific Paper

Performance Measurement of Complex Event Platforms

Eva Zámečníková *izamecni@fit.vutbr.cz*
Faculty of Information Technology
Brno University of Technology, Brno, Czech Republic

Jitka Kreslíková *kreslika@fit.vutbr.cz*
Faculty of Information Technology
Brno University of Technology, Brno, Czech Republic

Abstract

The aim of this paper is to find and compare existing solutions of complex event processing platforms (CEP). CEP platforms generally serve for processing and/or predicting of high frequency data. We intend to use CEP platform for processing of complex time series and integrate a solution for newly proposed method of decision making. The decision making process will be described by formal grammar. As there are lots of CEP solutions we will take the following characteristics under consideration - the processing in real time, possibility of processing of high volume data from multiple sources, platform independence, platform allowing integration with user solution and open license. At first we will talk about existing CEP tools and their specific way of use in praxis. Then we will mention the design of method for formalization of business rules used for decision making. Afterwards, we focus on two platforms which seem to be the best fit for integration of our solution and we will list the main pros and cons of each approach. Next part is devoted to benchmark platforms for CEP. Final part is devoted to experimental measurements of platform with integrated method for decision support.
Keywords: CEP, high frequency data, decision making, StreamBase, Esper, benchmarking.

1. Introduction

Our work is focused on a design of a method for formalization of business rules used during decision making process. This process is used by complex event platform (CEP) for better prediction of data. We would like to use the method in order to speed up the decision making process during time series prediction. For our solution we would like to use an existing solution of complex event processing platform where it is possible to implement our own module. CEP will ensure the prediction of data and we will add the component for decision making. As there are plenty of solutions we would like to find a solution which satisfies conditions for integration of our own module.

Nowadays there exists a number of complex event processing platforms. In general these platforms are a set of tools for the support of the preprocessing, processing and predicting of complex events. These platforms are designed for processing of data from multiple different sources and primarily focus on processing of moving data streams in real time. These data are processed on several levels of abstraction according to the required level of interference. The output of the process is pattern recognition, mining of trends and patterns in data and so predicting the flow of next input data.

Beside these platforms there are other tools supporting complex event processing such as frameworks, libraries, modules, etc. The current CEP tools do not solve identical problems so it depends on what purpose user wants to use these tools. Tools for CEP can be divided according to the characteristics of data. We would like to focus on tools which are designed for processing of high frequency data. These data occur in very small time intervals from multiple sources and in high volume. These data can be labeled as high frequency time series, are very variable and it is not possible to process them by using traditional approaches like linear models in statistics. They need to be described by nonlinear models. CEP has been used for various purposes like fraud detection, algorithmic trading, supply-chain monitoring, network management, traffic monitoring, call monitoring etc. CEP is often used in combination with service-oriented architectures (SOA). Information about CEP is based on [9].

1.1. Outline

Structure of the paper is as follows: After the introduction there is given a classification of CEP platforms and a brief overview of existing CEP frameworks. Afterwards a closer look is devoted to the decision making process in CEP. The new method for formal description of business rules set is introduced in this chapter. In the end of this topic the description of decision component implementation and the classification of business rules is given. This section is followed by the description of two CEP platforms. Two platforms were chosen on the basis of several parameters and these questions:

- How well can the platform process complex events?
- What is the speed of events processing on given platform?
- What is the latency of the system?
- How well is the decision making implemented on platform?

According to these questions we picked two solutions from nowadays existing solutions. Parameters that were determined for the comparison are listed after the description of platforms. It is followed by the comparison of parameters for both platforms with respect to the possibility of adding own module.

A framework for benchmark testing of CEP solutions is described at the end of the paper. This section if followed by experiments and measurements over the historical set of data from forex market data.Forex (foreign exchange market or currency market) is a global decentralized market for the trading of currencies. Afterwards the conclusion is given and a future development is mentioned.

2. Complex event processing (CEP)

2.1. CEP platforms classification

CEP platforms can be divided according to the aim of the platform into two groups:

- *Aggregation-oriented CEP* which focuses on calculation of average number of events in given time bounded window.
- *Detection-oriented CEP* which allows mining of event patterns and therefore predict possible opportunities or threats.

2.2. CEP frameworks

Currently most used CEP tools will be listed and described in the following text.

- *Sqlstream* - tool supporting automated actions from streaming analytics, possible integration with CEP platform (eg. SQLstream Blaze or Apache Hadoop -- open source framework that provides processing of high frequency data [16].
- *Microsoft StreamInsight* - commercial solution of CEP, includes the engine for decision making process, allows analysis of data in real time - supports monitoring, managing, and mining of the data for conditions, opportunities, and threats [12].
- *TIBCO StreamBase* - commercial solution, currently considered as the most complex tool for CEP with strong community support. StreamBase Component Exchange (SBX) is the community for StreamBase that allows users to download and distribute reusable components. This CEP solution is modular, it is possible to create user module and integrate it into existing solution [17].
- *Esper* - widely used open source solution, modular and allows processing of high volume data and event series ana lysis, available for Java as Esper, and for .NET as NEsper [5].
- *Oracle Stream Explorer* - part of Oracle Fusion Middleware -- open source software for parallel event processing, highly scalable, it's response time vary according to the volume of input data so it is not suitable for processing of high frequency data [13].
- *Others* - Coral8 - Sybase [10], SAP ESP [8], Apama [15], Apache Storm [2], etc.

From these solutions we picked two platforms - TIBCO StreamBase and Esper. Both solutions are widely used, have a strong community background and allow user to add his own modules. In the next sections we will discuss both approaches, and make a conclusion. We also chose them because of their stable position among CEP platforms. These platforms are not new - they exist over decade and they are still developing new features and improve overall solution.

3. Decision making process in CEP

A decision making process in CEP is implemented as stateful. This means that the decisions are not based just on the actual data that come to a system but historical sets of data are also taken into account. Decisions depend on other parameters like context of events, time, etc. CEP deals with relations between events of different situation types and thus can determine assessments and trends in data. The decision making engine uses predefined rules to identify situations. Rules can be captured by using EPL language which is designed for pattern description. Figure 1 shows schema of the decision making process in CEP. This schema is based on StreamBase CEP model.

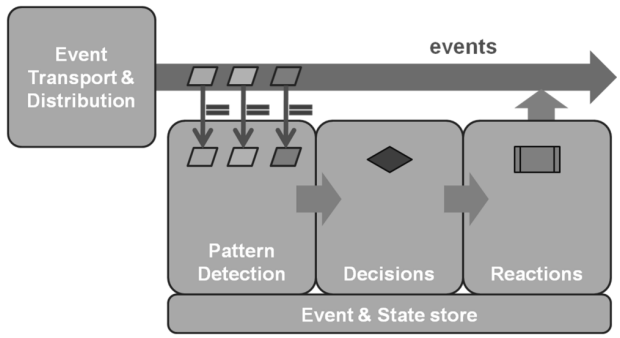

Figure 1Decision making process schema [18]

3.1. Business Rules

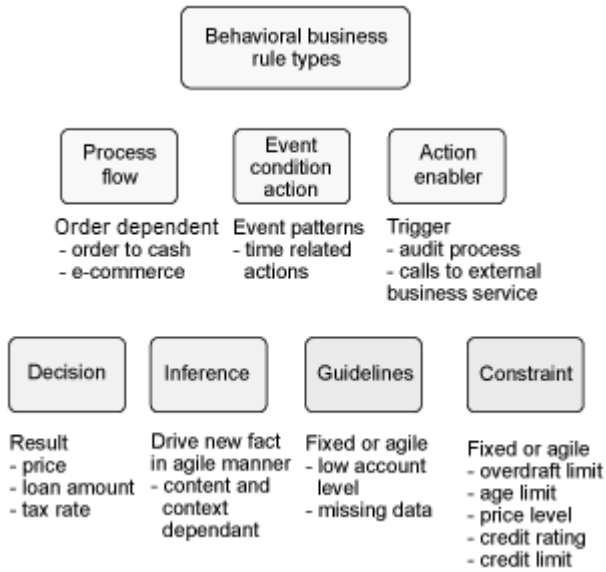

Figure 2 Business rules types [3].

Business rule approach manages the flow of business process by using constraints and/or decision blocks. Business rules classify, compute, compare and control data to direct their flow. Business rule patterns can simulate several types of events behavior such as logical operations, threshold patterns, subset selection patterns, modal patterns - check if assertion is true, time or spatial restrictions - according to the spatial restriction possible fraud can be detected. Business rules can be more structured and detailed statement, eg. condition-action statements. Single rule statement can yield more condition-action rules.

A classification of behavioral business rule types is presented in Figure 2. Behavioral rules can be further decomposed to support different patterns of implementation, depending on the granularity of the process implementation. Behavioral business rules express constraints or guidelines. Colors in figure mark different categories of business rules [6].

3.2. Formal model of decision making

In the first part of decision making process we recognize patterns then we make decisions and react to them. In the second step we will use the set of business rules for the decision making. This set contains the business rules which affect further processing of event flows and enable adding newly recognized patterns and rules. This should be done automatically in real time when the process is still running.
At this point we focus on the set of rules. We want to formally describe the set of business rules by matrix grammar and the dependencies between the rules will be represented by matrices of rules. Matrices allow us to model restrictions of the business process. In the step of processing other tools supporting decision making can be used eg. decision tables, vocabulary support.

3.3. Formalization of business rules

Formal grammars can be used for description of behavioral patterns and set of business rules extracted by CEP and for prediction of data in CEP platforms. Briefly, a formal grammar is a set of rules for rewriting strings, along with a "start symbol" from which rewriting starts. Matrix grammar belongs to the group of regulated rewriting grammars. For further reading about this topic authors recommend (Rozenberg et. al, 1997 [14]).

3.4. Definition of matrix grammar

Matrix grammar is a pair $H = (G, M)$, where $G = (N, T, P, S)$ is context-free grammar and M is finite language over P, $(M \subset P^*)$ - sentence of this language is called **matrix**.
Formally, a matrix grammar is a pair $H = (G, M)$, where

- $G = (N, T, P, S)$ is a context-free grammar, where:
  - N is an alphabet of nonterminal symbols

- T is an alphabet of terminal symbols
- P is a finite set of rules, $P \subseteq N \times (N \cup T)^*$
- S is starting symbol, $S \in N$

- M is a finite language over P, ($M \subset P^*$) - a sentence of this language is called a matrix.

Further, for $u, v \in (N \cup T)^*, m = p_1 \dots p_n \in M$ we define $u \Rightarrow v\,[m]$ in H, if there are strings $x_0 \dots, x_n$such that $u = x_0, v = x_n$, and for all $0 \leq i < n, x_i \Rightarrow x_{i+1}\,[p_{i+1}]$ in G. The language generated by H, denoted by $L(H)$, is defined as $L(H) = \{w: w \in T^*, S \Rightarrow^* w\}$.

Even thought that matrices contain only *context-free* rules, they may generate the *context-sensitive* language.

3.5. Formalization of Business Rules by Using Matrix Grammar

Input: Business rules in various forms. Business rules can be in the form of decision tables, enumeration of condition-action rules or sentences of natural languages. Form of business rules is discussed above. Rules are given in form of condition-action statements which are grouped into the matrices.

Output: DSS described by the business rules in the form of matrix grammar $H = (G, M), G$ is quadruple (N, T, P, S)

Method: $G = (N, T, P, S)$, where:

$N := \{Action_1, Action_2, \dots, Action_n\}$

$T := \{condition_1, condition_2, \dots, condition_m, action_1, action_2, \dots, action_n\}$

$P := N \times (N \cup T)^*$

for each $Rule_p$, $\{Rule_1, Rule_2, \dots, Rule_p\}$ from the decision table consider all suffice conditions, the set $\{Condition_1, Condition_2, \dots, Condition_m\}$ and do:

1. add rule $p, p \in P: S \rightarrow < Rule_1, Condition_1 >< Rule_1, Condition_2 > \cdots < Rule_1, Condition_m >$
2. add rule $p, p \in P: S \rightarrow < Rule_2, Condition_1 >< Rule_2, Condition_2 > \cdots < Rule_2, Condition_m >$
3. ...
4. add rule $p, p \in P: S \rightarrow < Rule_p, Condition_1 >< Rule_p, Condition_2 > \cdots < Rule_p, Condition_m >$,
5. add $< Rule_p, Condition_m >$ to N; m, p are positive integers.

For each $< Rule_p, Condition_1 >$ add rules:

6. $< Rule_p, Condition_1 > \rightarrow Action_n condition_1$
7. $Action_n \rightarrow action_1 action_2 \dots action_n$, where $action_n$ are all actions taken after fulfilling of all sufficient conditions for the $Rule_p$.

For each $< Rule_p, Condition_m >$for $m \in N, m > 1$add rules:

- $< Rule_p, Condition_m > \rightarrow condition_m$

$S := S;$
$M := \{\mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_\mathrm{p}\}$, where $\mathrm{m}_\mathrm{p} = [< Rule_p, Condition_1 >$
$\rightarrow Action_n condition_1, < Rule_p, Condition_m >$
$\rightarrow condition_m$ for all $\mathrm{m} > 1]$

Component M is usually created by business analyst by determining parallel actions. Only the actions that leads to the execution of actions are added to matrices. In this case the matrices are determined by grouping of all conditions into matrix and all actions in one matrix. All rules in each matrix have to be taken in one computational step.

It is intended to implement decision making system by using decision service and SOA. Decision services may not be linked only to business process activities. The majority of decision services deployed in SOA are not directly linked to any automated business process. We need to keep the set of business rules in separated component so we can maintain it by adding new rules, removing or updating current business rules. This is followed by traditional SOA approach, where service identification, specification and implementation is done to address reusability, adaptability, and change management needs. Based on results published in [19].

4. (N)Esper platform

Esper is an open source engine that combines both CEP approach and event stream processing (ESP). ESP queries involve simple select queries and window aggregations on a single stream of data. CEP is a super set of ESP. Differences between ESP and CEP are discussed in [9]. In CEP, we find patterns, derive new events based on a combination of input events, possibly from multiple streams of data. Esper is available in Java or in C# .NET as NEsper. This platform enables rapid development of applications that analyze high frequency data, combining historical and real-time data. Esper filters and analyzes events in various ways and responds to conditions of interest. Esper provides a rich declarative language for dealing with high frequency time-based event data for pattern definition called Event Pattern Language (EPL). EPL is SQL based and offers all SQL operators extended with temporal operators. Spatiotemporal patterns are defined in the ESPER knowledge base pattern and they are used by the pattern matching process. The goal of CEP is to identify meaningful events (opportunities or threats) and respond to them as quickly as possible [4].

4.1. Applications using Esper

Examples of applications using Esper are:

- Business process management and automation (process monitoring, BAM, reporting exceptions, operational intelligence).
- Financial instruments (algorithmic trading, fraud detection, risk management).

- Network and application monitoring (intrusion detection, SLA monitoring).
- Sensor network applications (RFID reading, scheduling and control of fabrication lines, air traffic).

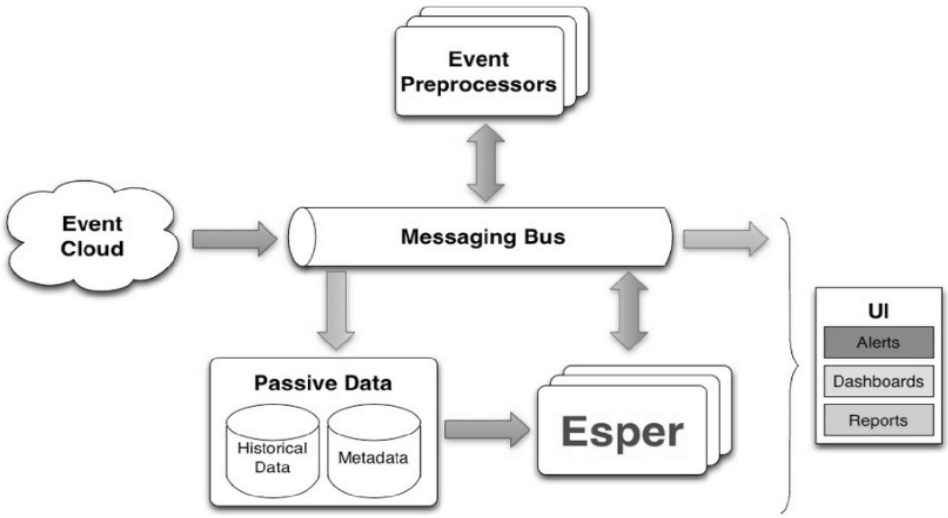

Figure 3 Schema of Esper platform [1]

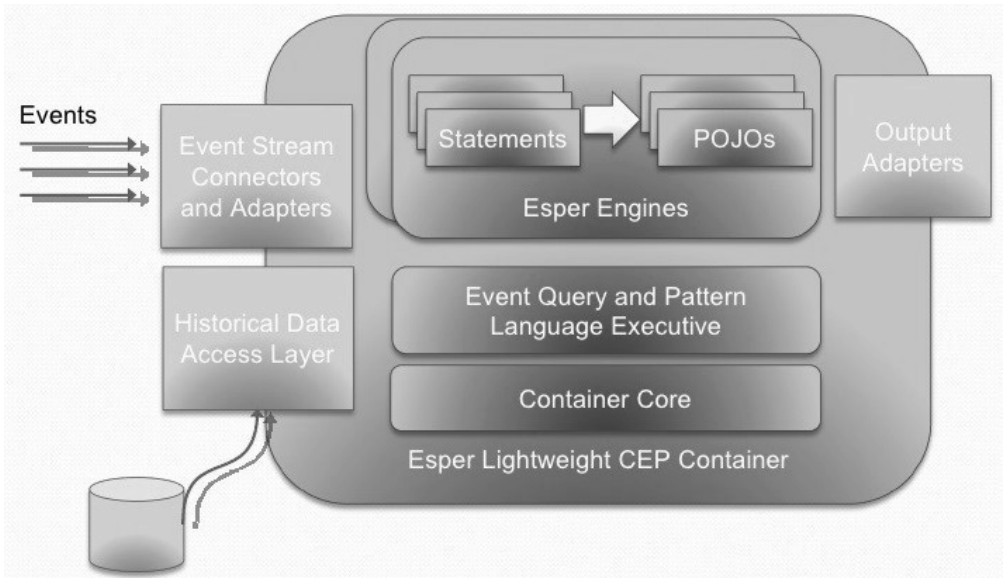

Figure 4 Core of Esper CEP platform [9]

In Figure 3 schema of Esper platform is displayed and a core of Esper engine is in Figure 4. The engine of Esper is based on the use of state machine technology. We find this feature interesting and quite simple for integration - in

comparison with other tools - with the model of set of business rules controlled by matrix grammar. Esper includes a historical data access layer to connect to the most of the common databases and it is also possible to combine historical data and real time data in one single query. Esper can be easily integrated with most available servers (Weblogic, Websphere, JBoss, Tomcat, etc.), service buses, grid platforms, and Microsoft based .Net technologies for NEsper. This platform supports different kinds of input event formats, from Java / .Net objects and maps to XML documents. Esper engine includes failover and recovery capabilities, ensuring that the engine is non-stop usable (high-availability). Another advantage is custom adding of event storage options. As performance tests show Esper scales vertically nearly linearly (adding more CPU power). In a VWAP (Volume Weighted Average) benchmark. Esper exceeded 500.000 events per second on a dual CPU server class hardware, with only 5 microsecond average latency. Horizontal scaling is best handled by logical partitioning of statements and data streams to separate Esper instances [4].

Esper offers work with time-based batching window, for example, combining events for specific time window size (1min, 30seconds, etc.). This feature is very important for the decision making process for example for detecting of threats. For example, if events can be batched for the previous 1 minute and a fault can be found within this time window it can be predicted immediately. For a real life problem, the size of time window needs to be set very precisely. The Esper CEP maintains a batch buffer to keep all the events coming into the Esper [1]. Batch buffers also serve as means to cope with network distribution issues: business platform that generates a lot of events that need to be consumed by many clients might choose to group these events by a time unit to keep the network stress level low, instead of distributing these events one by one.

Esper's advantage is that it is open-source software. In comparison with other CEP it doesn't have as many tools as eg. StreamBase provides, but its strenght is in the core engine that is embeddable into third-party solutions.

5. StreamBase CEP platform

According to the Forrester Research [5] ,which is evaluation of customer relationship management, StreamBase CEP platform is a leader among today's CEP platforms. This platform is set up from several tools such as server, IDE, connectivity adapters that create complex platform for preprocessing, executing and predicting of input data. It is a software for rapidly building systems that analyze and act on real-time streaming data. StreamBase combines an application development environment, an event server with low-latency high-throughput, and enterprise connectivity to real-time and historical data.

StreamBase uses graphical language EventFlow, it can compile multiple EventFlow or StreamSQL queries at run time. StreamBase is the only CEP in the industry that uses visual language for application development. This visual-based language gets compiled into low-level code before execution. StreamSql, according to its name, is a query language that enables the processing of real-time

data streams. StreamBase engine uses in-memory cache so the in-memory databases run faster than traditional relational databases.

5.1. Applications using StreamBase

Examples of applications using StreamBase are:

- Intelligence and Security (fraud detection, military purposes).
- Capital Markets (algorithmic trading, Market Data Management, smart order routing).
- Retail, Internet and Mobile Commerce (retail promotion, website monitoring).

Telecommunications and Networking (network monitoring and protection, fraud detection, bandwidth and Quality-of-Service Monitoring).

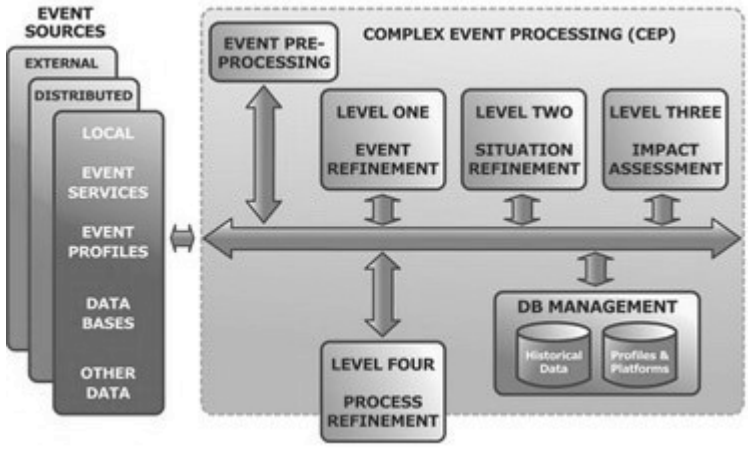

Figure 5 Schema of StreamBase CEP platform [9]

As Figure 5 shows, CEP is composed of several levels which conform to desired level of inference. The event preprocessing runs at the lowest level. During this phase the input data streams are cleaned to produce some understandable data. On the next level, the events that were detected in input data are refined and subsequently initial decisions and correlations are done. The main challenge is to find relevant data. Then situation refinement and impact assessment follows. At the level of impact assessment, we may predict the intentions of subject or to estimate potential losses or opportunities. At the end, the process refinement is done. All the results of event processing and operational visualization at all levels are summed up in a human readable format via user interface.

StreamBase incorporates Java, C++ and Python into all StreamBase applications. This platform is modular, users may integrate their own solutions into existing platforms and thanks to the community StreamBase Component Exchange (SBX) it is allowed for users to download and distribute reusable components. In this case the integration of decision making model can be implemented as a single module. One thing which may seem to be a disadvantage,

in comparison with Esper, is that this software is not freely available. This section was based on information available at [17].

For the purpose of benchmark testing StreamBase fuses actual and historical data. Historical contextualization of real time data ensures better decisions. Looking at historical algorithmic decisions can highlight changes in client behavior that may represent opportunities for relationship building.

6. Comparison of CEP platforms

For the purpose of integration of method for decision making to TIBCO StreamBase or Esper we focused on several characteristics of each platform:

- *Modularity* Both platforms support the integration of a custom module.
- *Decision making engine* The presence of a decision making module in TIBCO StreamBase is an advantage as we can implement proposed method and compare the results with original solution.
- *Pattern matching* Both platforms support the addition of user queries. Esper has its own event processing language EPL with SQL-like syntax. StreamBase offers query operator for defining user queries.
- *Batch window size* StreamBase allows user to define the batch window size according to the several parameters - the volume of events, time window or field based window which uses events whose values falls within a certain range. Esper provides the length and time based window size.
- *Benchmark testing* As previously stated Esper scales vertically nearly linearly when adding new CPU. StreamBase is battle-tested for low latency and real-time risk management, for vertical scaling the StreamBase monitoring utilities can evaluate the hot spots and distribute the computing into more threads for better performance.

For the purpose of the use of platform for high frequency time series prediction few parameters are crucial. For the processing of high frequency data we need to be able to process these data and to have the response from system in near real time. From this point of view we found following parameters for benchmark tests and their characteristics interesting:

- *Latency* Latency is the lag between detection of two complex events in the set of triggering events sent to the CEP engine. In our setup we note the time in milliseconds before sending each event. Upon matching a statement the updateListener function would be invoked with the events. There we update the stats module with the current time - last event time.
- *Throughput* Throughput is the maximum number of events per second which the CEP engine can process without loss of data or without clogging the queues. The current setup uses a channel which blocks input on the application level if the channel buffers are full. So the client program will not be able to write data to the

channel any faster than the server consumes it. At 100 % CPU utilization, the throughput may decrease a little and the latency may increase.

- *CPU Utilization* It is the CPU Utilization for different kinds of CEP query over different event rates for a given pattern.
- *Memory Utilization* It is the memory profile for different kinds of CEP query over different event rates for a given pattern.

According to the Esper specification Esper exceeds over 500 000 event/s on a dual CPU 2GHz Intel based hardware, with engine latency below 3 microseconds average (below 10 microseconds with more than 99 % predictability) on a VWAP benchmark with 1000 statements registered in the system - this tops at 70 Mbit/s at 85 % CPU usage. Esper also demonstrates linear scalability from 100 000 to 500 000 event/s on this hardware, with consistent results accross different statements [4].

StreamBase baseline quad-core machine handles 140,000 input messages per second with latency of 86 microseconds. Scaling to an eight-core AMD machine, the StreamBase processes 245,000 input messages per second, with 71 microseconds of latency, demonstrating a scaling factor of 0.875 across multicore architectures. In cooperation with AMD, on the 8-core machine, throughput was 245,400 input messages per second. The breakdown was 207,800 market data updates per second and 37,600 orders per second with lower latency [18].

7. CEP Benchmark Testing frameworks

Nowadays, several solutions exist for measuring the performance of CEP platforms. Most of them started as a university project - FINCoS, BiCEP, CEPBen. We will describe the first of them as it is more complex and flexible than others. The idea of all three is basically the same.

7.1. FINCOS

According to the [11] FINCoS is a set of benchmarking tools for load generation and performance measuring of various event processing systems. It allows to create synthetic workloads and enables to evaluate candidate solutions using user's own datasets. An extensible set of adapters allows the framework to communicate with different CEP engines and its architecture permits to distribute load generation across multiple nodes.

The FINCoS framework is composed by five main components:

- *Drivers* - simulate external event sources, submitting load to the system under test.
- *Systemundertest* - tested CEP engine. The results produced by the system under test are received and stored in log files for subsequent answer validation and performance measurement.
- *Sinks* - receive the results produced by system under test.

- *Adapters, Controller* - the communication with the CEP engine is made through an extensible set of adapters. Controller allows users to configure, execute, and monitor performance tests through GUI.
- *Performance Monitor component*- the results of performance tests can then be visualized both in real-time and after test completion, using the Performance Monitor component.

The execution of drivers can be split into phases, each with its own workload characteristics. This is useful not only for breaking performance tests into well-described parts, but also for evaluating the ability of event processing platforms in adapting to changes in the load conditions. In addition, users can choose if events should be generated by the framework itself or read from files containing real-world event data.

The workload can also be seamlessly scaled by simply adding more drivers and sinks to the configuration. The framework supports direct communication with event processing platforms through custom adapters.

8. Measurements and testing of decision making module

Decision making module implemented and integrated to Esper platform is based on EPL. The patterns in Esper takes form of SQL-like declarative rules that are given to the engine in the form of decompiled String, e.g.:

```
String epl = "select tick.price as tickPrice,
trade.price as tradePrice, sum(tick.price) +
sum(trade.price) as total
  from pattern [every tick=StockTickEvent or every
trade=TradeEvent].win:time(30 sec)";
EPStatement statement =
epService.getEPAdministrator().createEPL(epl);
```

Pattern syntax in Esper is done by using pattern statements. Pattern statements are created via the `EPAdministrator` interface. The `EPAdministrator` interface allows to create pattern statements in two ways:

- Pattern statements that want to make use of the EPL select clause or
- other EPL constructs use the createEPL method to create a statement that specifies one or more pattern expressions.

Use the syntax is shown below.

```
EPAdministrator admin =
EPServiceProviderManager.getDefaultProvider().getEPAdmi
nistrator();

String eventName = ServiceMeasurement.class.getName();
```

```
EPStatement myTrigger = admin.createEPL("select * from
pattern [" +
     "every (spike=" + eventName + "(latency>20000) or
error=" + eventName + "(success=false))]");
```

All measurements were performed by using quad core 64-bit Operating System with Windows 7, 7-4600M CPU @ 290GHz 290 GHz Intel based hardware with 16GB RAM.

Information about EPL and pattern syntax based on [4].

8.1. Test cases

We measure the latency,throughput,CPU and memory utilization for our integrated solution. More detailed information about test cases is summarized in following subsections.

8.1.1. Latency

Latency is a significant user metric in many real-time applications. Users are usually interested in quantiles of latency, such as worst case or 99th percentile. Measurement proved that latency of system was below 3microseconds for 99%.

8.1.2. Throughput

Throughput is expressed in events/s. Experimental measurements proved that throughput ranged from 150000 to 200000 events processed per second. The measurement was performed no longer than 10 min after startup.

8.1.3. CPU Utilization

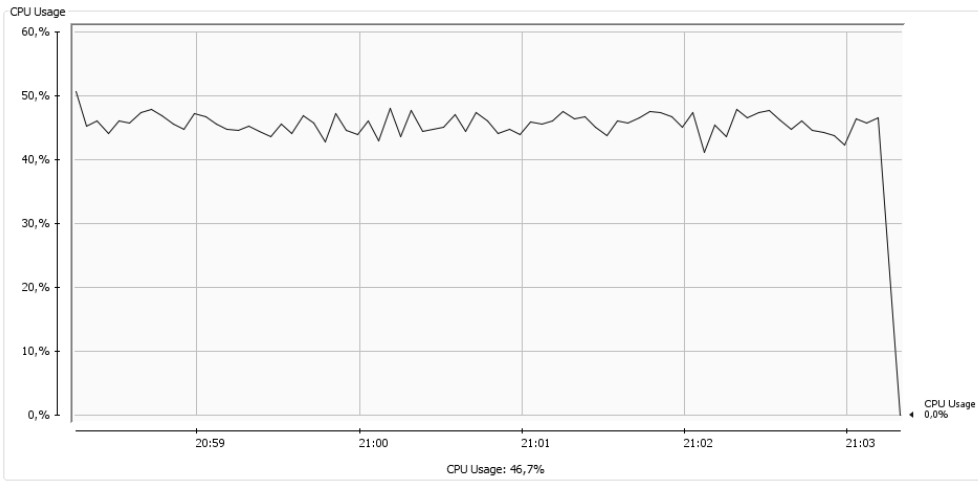

Figure 6 CPU utilization

CPU utilization was measured within the range of 5 minutes interval. The applied load and the CPU usage correlated. The memory consumption was almost constant. The average count of threads which were processing at each moment was 16. The measurement was performed no longer than 10 min after startup. In figure 6 is captured the CPU utilization within 5 minutes.

8.1.4. Memory Utilization

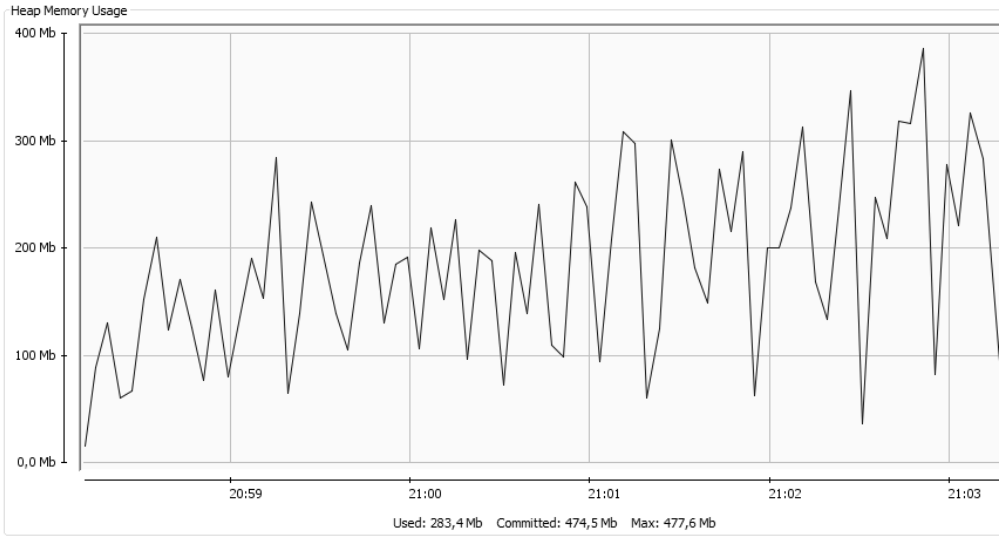

Figure 7 Memory heap utilization

Memory utilization was measured within the range of 5 minutes interval. The average count of threads which were processing at each moment was 16. Memory heap utilization ranged between 100Mb to350Mb of used memory. The measurement was performed no longer than 10 min after startup. In figure 7 is captured the CPU utilization within 5 minutes.

Conclusion

This paper discusses the CEP platforms for high frequency data processing and compares two solutions which allow user to add custom module into existing platform. At the beginning brief overview of existing CEP tools is given and two solutions are described in more detail. We aimed on those two platforms as they already have a component for decision making. By adding of our own component we can make experimental measurements and compare new decision making system with the original one. Decision making system is based on business rules definition by using a formal grammar. Similar approach was already described in [7].

Two compared platforms were chosen according to the requirements which need to be fulfilled before we integrate an implementation of decision making process controlled by formal grammar component.

Both platforms have been among the best CEP platforms for a long time.

The main advantage of the use of the TIBCO StreamBase for our solution of decision making component is its modularity. On the other hand this platform is not free to use. The advantage of Esper solution is free license and the existence of Esper expression language. The disadvantage against the StreamBase is an absence of decision making engine, so there is no way to run experiments and compare the results with the original solution.

8.2. Future work

Future work will be to test implemented method of decision on more input data from different sources. The implementation of decision making system will be run on historical set of data and the prediction of data will be compared to original solution. The main purpose of implementation of our own decision making system is to fully describe the business rules by formal model. As a formal model we chose the matrix grammar as it allows to model restrictions of actions upon the data and partly can simulate the parallel processing of actions in scope of business process. The implementation of this approach can be used for the formal verification of CEP systems. This area is still not fully explored.

Acknowledgements

This work was partially supported by the BUT FIT grant FIT-S-14-2299, "Research and application of advanced methods in ICT".

References

[1] A. Alegria. (2011, November). *Complex Event Processing with Esper*, [Online]. Available: http://www.slideshare.net/antonio_alegria/complex-event-processing-with-esper-10122384. Accessed: Jun. 28, 2016.

[2] A. S. Foundation. (2015). *Apache Storm* [Online]. Available: http://storm.apache.org/. Accessed: Jun. 28, 2016.

[3] J. Boyer, S. T. S. Member, and I. China. (2012). *Best practices for designing and implementing decision services, part 1: An SOA approach to creating reusable decision services* [Online]. Available: http://www.ibm.com/developerworks/bpm/bpmjournal/1206_boyer/1206_boyer.html. Accessed: Jun. 28, 2016.

[4] Esper. (2016). *Esper* [Online]. Available: http://www.espertech.com/esper/. Accessed: Jun. 28, 2016.

[5] Forrester Research. (2016). *Create the great customer experiences that grow revenue* [Online]. Available: https://www.forrester.com/home/. Accessed: Jun. 28, 2016.

[6] B. von Halle and L. Goldberg, Eds., "The business rule revolution: Running business the right way"; Cupertino, CA: Happy About, 2006.

[7] R. Hypský, E. Zámečníková, J. Kreslíková, "Formal Definition of Business Rules by Grammar Systems", International Journal of Advancements in Communication Technologies (IJACT). Rome, 2015, ISBN 978-1-63248-044-6. ISSN 0976-5697.

[8] *Introduction to SAP event stream processor* [Online]. Available: http://scn.sap.com/docs/DOC-46304. Accessed: Jun. 28, 2016.

[9] D. C. Luckham, "Event processing for business: Organizing the real time enterprise". United Kingdom: Wiley, John & Sons, 2011.

[10] MDS App Tech. (2016). *SAP gold partner and leading SI for BI/Analytics, data management and ERP solutions* [Online]. Available: http://www.sybaseproducts.com/. Accessed: Jun. 28, 2016.

[11] N. Mendes,R. Marcelo, P. Bizarro, P. Marques, "FINCoS: Benchmark Tools for Event Processing Systems", 2014.

[12] Microsoft. (2016). *Microsoft StreamInsight* [Online]. Available: https://technet.microsoft.com/en-us/library/ee362541(v=sql.111).aspx. Accessed: Jun. 28, 2016.

[13] Oracle. *Oracle complex event processing* [Online]. Available: http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html. Accessed: Jun. 28, 2016.

[14] G. Rozenberg and A. Salomaa, Eds., "Handbook of formal languages: V. 2: Linear Modelling - background and application". Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 1997.

[15] Software AG. (2016). *Apama streaming Analytics - complex event processing.* [Online]. Available: http://www.softwareag.com/corporate/products/apama_webmethods/analytics/overview/default.asp. Accessed: Jun. 28, 2016.

[16] Sq. Incorporated. (2010). *Real-time processing for big data in motion* [Online]. Available: http://www.sqlstream.com/. Accessed: Jun. 28, 2016.

[17] T. S. Inc. (2016). *Event processing* [Online]. Available: http://www.tibco.com/products/event-processing/. Accessed: Jun. 28, 2016.

[18] P. Vincent, The TIBCO Blog. (2009). *CEP: More than event patterns* [Online]. Available: http://www.tibco.com/blog/2009/12/18/cep-more-than-event-patterns/. Accessed: Jun. 28, 2016.

[19] E. Zámečníková, J. Kreslíková. "Comparison of Platforms For High Frequency Data Processing". In: 2015 IEEE 13th International Scientific Conference on Informatics. Poprad, SK: Technická univerzita v Košiciach, 2015, s. 296-301. ISBN 978-1-4673-9867-1.