# A Visualization System for High Dimensional Data Streams using Complex Event Processing

Arnab Chakrabarti*
RWTH Aachen University, Germany

Tanuj Kulshrestha†
RWTH Aachen University, Germany

Christoph Quix‡
Hochschule Niederrhein, Germany
Fraunhofer FIT, Germany

**ABSTRACT**

Real-time streaming data analysis and visualization are of main interest for businesses nowadays to quickly take important decisions that could help them to enhance the decision-making strategies. Visualization of real-time data streams helps in the exploration of evolving data. High-dimensional data is difficult to visualize and interpret due to the limitations of the display screen that lead to visual clutter. We propose a system which can efficiently perform dynamic feature selection using complex event processing with no prior knowledge of the features. Our system renders data related to a selected number of features, rather than processing and visualizing the entire data frame, thus reducing the memory consumption and increasing performance. We present a distributed architecture of a framework which helps in the exploration of high dimensional data streams using different plots and charts in a dashboard that updates the data in real-time.

**Index Terms:** Human-centered computing—Visualization—Visualization Framework—High Dimensional Data Visualization; Feature Engineering— Data Stream Visualization—Feature Selection—Complex Event Processing

## 1 INTRODUCTION

In recent times, an enormous amount of data is being generated by various heterogeneous sources and mostly the data is multidimensional making it complex and difficult to interpret. One of the proven methods to efficiently communicate, comprehend and interact with this complex and large amounts of information is data visualization. However, the increasing dimensionality and the growing volumes of the data pose a challenge to the current analytic systems to visualize high dimensional data and unfold the hidden information. This is mainly because human cognition limits the number of data dimensions that can be visually interpreted. The potential amount of overlapping data points projected on to a two-dimensional display hinders the interpretation of meaningful patterns in the data. Similarly, some business requirements need to handle real-time complex events from high dimensional data streams. Visualizing the trends from these data streams enables stakeholders to make effective decisions. For example, in case of weather data, if forecast data shows an upward trend for temperature or precipitation then it generates an alert for farmers to take timely action for the crops or monitoring trends in wind speed and direction could help in real-time actions for flights by the air traffic controller. Visual rendering of this high dimensional data in real-time requires high utilization of CPU and memory. Recently, there have been a lot of developments on the reduction of high-dimensional streaming data using the state-of-the-art in dimensionality-reduction techniques. Despite the adoption of various strategies, information loss is still a major challenge which makes many visualization strategies insufficient. To deal with the visualization issues arising from big multidimensional data, we propose a real-time visual analytics system that is capable of handling the issues related to scalability and high dimensionality. We handle the "curse of dimensionality" by proposing a novel feature-selection strategy using *Complex Event Processing (CEP)* which detects features that are useful and important for business goals. We also present a distributed architecture of our proposed system using an integrated Apache Storm-CEP topology thus enabling us to seamlessly ingest and visualize high dimensional data streams and identify hidden data trends with minimum latency.

### 1.1 Identifying Data Trends with CEP

Complex Event Processing(CEP) [6] is an efficient technique for processing high-velocity data in real time. CEP helps in the analysis of large flows of primitive events to timely detect situations of interest, patterns or trends [4]. In CEP, processing takes place according to user-defined rules or queries that detect events and patterns from the event streams. These rules can be written beforehand in the rule engine and executed to detect the patterns and complex events. It is also possible to update the rules in run-time by automatically adapting to the event patterns using Machine Learning and Data Mining techniques [4, 5]. A time-series is a series of data points indexed in time order, and a streaming time-series $S$ is a sequence of real values $s_1, s_2, \ldots, s_n$, where new values are continuously appended as time progresses [3]. Detecting trends requires us to analyze event streaming data which is a sequence of data coming at different time stamps. Events are single atomic occurrences of interest at a given point in time. Based on the semantics and content of events they are categorized as primitive event and composite (or complex) events. Primitive events are single atomic events that stream into CEP systems, while composite events are detected and extracted by CEP systems based on the rules having one or more primitive events combinations [2]. For instance, in real-time weather streams "increase of temperature" is one primitive event tuple $t$ of type $T$. Similarly, "increase in humidity" is another primitive event tuple $p$ of type $P$. In CEP systems a rule can be defined that "increase of temperature" and "increase of humidity" together generate an alert for a hot day. The latter is a composite or complex event defined by using both $T$ and $P$. As one of the main contributions of this work we present an approach that processes streaming data using CEP for feature selection and trend detection for high-dimensional time-series data streams.

## 2 OUR APPROACH

Visualizing high dimensional data leads to the problem of overplotting which makes hidden trends difficult to identify. Our approach in visualizing high dimensional data streams is based on the principle of data reduction using feature extraction through a CEP engine. The contribution of our work is two-fold. First, we introduce the use of a CEP based generic system that process the data with unknown features dynamically. Second, we present a scalable framework which supports interactive visualization and trend detection

*e-mail: chakrabarti@dbis.rwth-aachen.de
†e-mail: tanuj.kulshrestha@rwth-aachen.de
‡e-mail: christoph.quix@fit.fraunhofer.de

of big data. As CEP in itself is not distributive, it is combined with distributive stream processing architecture called Apache Storm to improve the performance of real-time visualizations.

### 2.1 Dynamic Feature Selection using CEP Engine

To explain the concept of dynamic feature selection we use a sample use case scenario of identifying trends in a high dimensional weather data consisting of multiple weather events each having different features like temperature, pressure, humidity, precipitation, cloud_cover, latitude, longitude and altitude to name a few. For complex event processing, we used ESPER [1] as the CEP engine that uses the query language called Event Processing Language (EPL) for processing and analyzing streams of data on the go. The basic idea of CEP is to filter or extract complex events from the event streamed data.

#### 2.1.1 Query Generation using CEP Engine

Once we receive the data stream we convert them into event stream by creating events in run-time using ESPER CEP engine. The example below shows a sample query that creates the event named as *"weatherEvent"*.

```
@EventRepresentation(objectarray)
CREATE SCHEMA weatherEvent as (prop1 Map)
```

In the next step, we select distinct features from the *"weatherEvent"* stream using a single-row function *"featureSelectionVarience"* that selects features if the variance of the features is not zero i.e the values of features changes with time and in turn removing the features which are stagnant and does not contribute in understanding data trends. As it is difficult to analyze the entire stream at once we use sliding windows of fixed lengths. Within this length window, we detect features that are changing with time using CEP queries. Below are the simple CEP queries examples in ESPER.

```
SELECT distinct featureSelectionVariance(e)
from weatherEvent.win:length_batch(5) as e
```

Then we execute the query "trendsDetection" which is a single-row function to dynamically detect the trends using Moving Average for all the features and add the features under the category of trends such as increasing, decreasing and turn. Below is the example query used for detecting trends using moving average.

```
SELECT distinct trendsDetectionMA(e) from
weatherTrendMAEvent.win:length_batch(5) as e
```

We present our proposed algorithm which is used for dynamic feature selection using variance (Algorithm 1) and detection of trends using the concept of moving average (Algorithm 2).

With Figure 1 we present the complete data flow starting from generating streams of data to visualization of data trends.

Whereas Figure 2 depicts the visualization of the selected features from the high-dimensional weather data using a "clutter free" parallel coordinate system. Thus we have been successfully able to select only those features which are meaningful to the analyst. We want to reiterate that the features are extracted dynamically based on the business logic which the analysts can model into dynamic CEP queries at run-time. In this particular use case, only those features from the weather data which are important for trends are selected, eliminating the features that are stagnant over time. In Figure 2 we can also see from the dynamic stacked bar chart that temperature, humidity, precipitation, pressure, and wind-speed are the features that keep on changing every 20 seconds showing upward or downward trends. We can clearly identify that wind-speed and temperature are showing an upward trend in 3 minutes time window whereas precipitation keeps on increasing and decreasing. In the later section we will present briefly the visualization dashboard that is developed as a part of our work to support real time feature extraction and data trend exploration.

**Algorithm 1:** Feature Selection using Variance

```
Procedure : featureSelectionVariance;
Input: Initialize the List of HashMaps lstMaps of all the
  events from the LengthWindow W;
Output: List of HashMaps outMaps with selected features;
Calculate average using streams and lambda functions;
for all the HashMaps mapper with key in List do
    mapper.put(key, Math.pow(average.get(key) -
      mapper.get(key), 2);
end
Calculate variance using streams and lambda functions;
for all the keys in HashMap variance do
    if varianceValue != 0 then
        map.put(keys, mapper.get(keys));
        outMaps.add(map);
    else
        continue;
    end
end
```

**Algorithm 2:** Trends Detection using Moving Average

```
Procedure : trendsDetectionMA;
Input: Initialize the List of HashMaps lstMaps of all the
  events from the LengthWindow W;
Output: trend as HashMap with keys rise, fall and turn;
Calculate average for consecutive HashMaps using streams
  and lambda functions;
for all the maps in lstMaps do
    for all the keys in firstMap do
        if (thirdvalue > secondvalue) and (firstvalue <
          secondvalue) then
            trend.put(rise, third);
        else if (thirdvalue < secondvalue) and (firstvalue >
          secondvalue) then
            trend.put(fall, third);
        else
            trend.put(turn, third);
        end
    end
end
```

## 3 System Architecture and Visualization Dashboard

To perform real-time visualization, there is a need for having a system that performs well in terms of latency, memory, and CPU consumption. It is important to process the data stream quickly without any delay and overhead to make visualizations responsive. Also, the goal of our work was to construct generic queries that perform feature selection without any prior knowledge about the features. To achieve that, complex event processing is used to dynamically identify features and in turn performs feature selection and trends detection. Extensive research has been done in selecting the stream processing framework that fulfills both the above-mentioned goals. But except ESPER, the complex event processing engine, none of the stream processing frameworks like Spark Streaming and Kafka Streaming can perform efficiently in processing unknown features for streaming data. These systems needs to know all the feature names in advance for aggregation, pattern-matching, and filtering. In terms of performance, ESPER has lower latency and

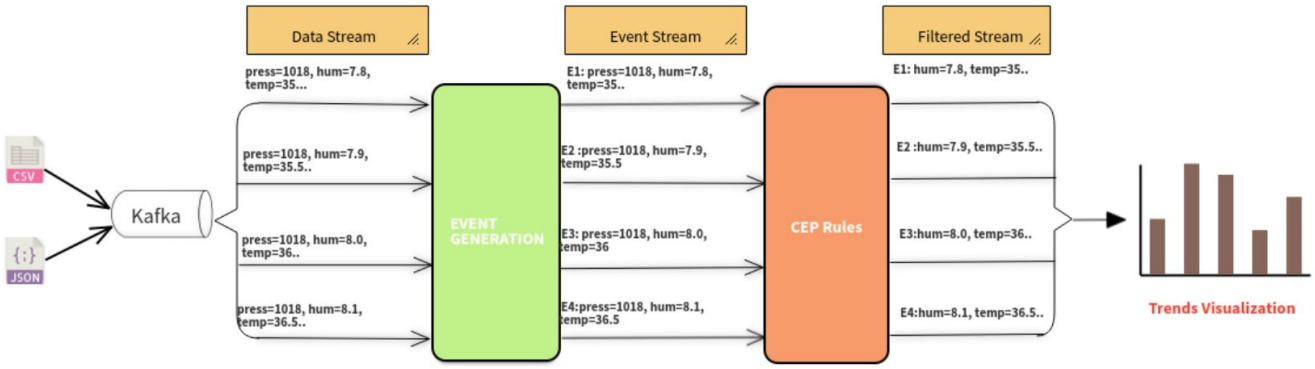


Figure 1: Feature Selection using CEP.

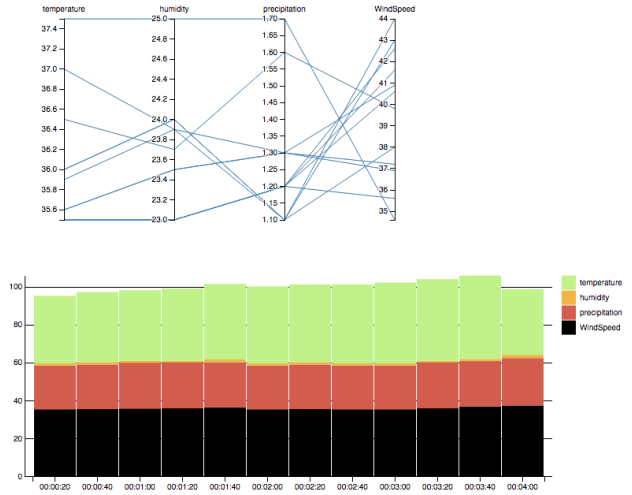


Figure 2: Visualizations of Selected Feature Set and their hidden trends

higher throughput as compared to other stream processing frameworks as discussed above which makes it more suitable for real-time visualization.

### 3.1 Storm-Esper Distributed Topology

For distributive processing and visualization of data streams, there is a need for a real-time stream processing framework that provides the best performance in terms of latency and throughput and also performs well in feature selection and trends detection. As trends detection is only meaningful on the features that are changing with time, two levels of data stream processing have been proposed in which at first features are selected by user defined CEP queries and then the resulting data streams are passed to the query engine for trends detection. For this kind of processing Apache Storm is useful which has a concept of spouts and bolts. In storm topology, spouts act as a source for streams that transforms the data taken from the message queue to tuples (key-value pairs) and send it to the bolts. Bolts are the processing unit of Storm topology that process the tuples. Apache Storm provides an option to rearrange bolts in any order according to the requirement. It is also possible to perform stream grouping to decide which streams go to which bolts and do load balancing to speed up the execution. In storm topology bolts receive streams of data in the form of tuples from storm spouts. For our work we use Apache Kafka as a spout in the topology that takes data streams from Kafka Topics and passes it to the bolts for processing. Kafka Spouts ingest data streams and convert them into tuples and pass it on to the feature-selection bolt in which CEP queries are running. Once the feature selection bolt processes the streams, it will pass the resulting tuples to the trends detection bolt which executes the CEP queries for trend detection.

The architecture as shown in Figure 3 consists of several components like streaming data sources, Apache Kafka, Apache Storm, ESPER CEP engine and InfluxDB. Our proposed system utilizes the components together to create a real-time streaming visualization framework to understand the evolution and trends of the data with time. The architecture consists of front-end and back-end modules. In the back-end at first Kafka is used for data pipeline that takes data from APIs and other streaming sources to process it further. Kafka producer sends the data synchronously to the topics and then consumer consumes the data from it. Storm architecture is used for distributive processing of the data that has the concept of spouts and bolts explained in the previous section. Here, Kafka spout is acting as Kafka consumer that consumes the data from the topics and transform it to the stream of tuples. These tuples are sent to the bolts where complex event processing takes place with CEP queries. The storm topology consists of one Kafka Spout, two feature selection bolts and trends detection bolt. Once the features are selected and trends are detected, the resulting streams are stored in the InfluxDB and visualized using the visualization dashboard.

Below in Figure 4 we present the complete visualization dashboard that we have developed for the purpose of this work. Due to the lack of space we are not able to describe the implementation details of the proposed framework however a detailed explanation can be found here: `http://dbis.rwth-aachen.de/cms/staff/chakrabarti/CEPSVIZ`

## 4 Evaluation

We have evaluated our approach over 6 data-streams (by connecting with data sources generating real time high dimensional data) to compare with a baseline approach (using Apache Flick), and demonstrate that our approach discovers meaningful feature sets with low latency and high throughput.

### 4.1 Experimental Setting

The implementation of proposed framework has been carried out using JAVA programming language, Zookeeper, Apache Kafka, Apache Storm, ESPER CEP engine, InfluxDB, and Grafana running on the remote Linux server. The server hosts a collection of

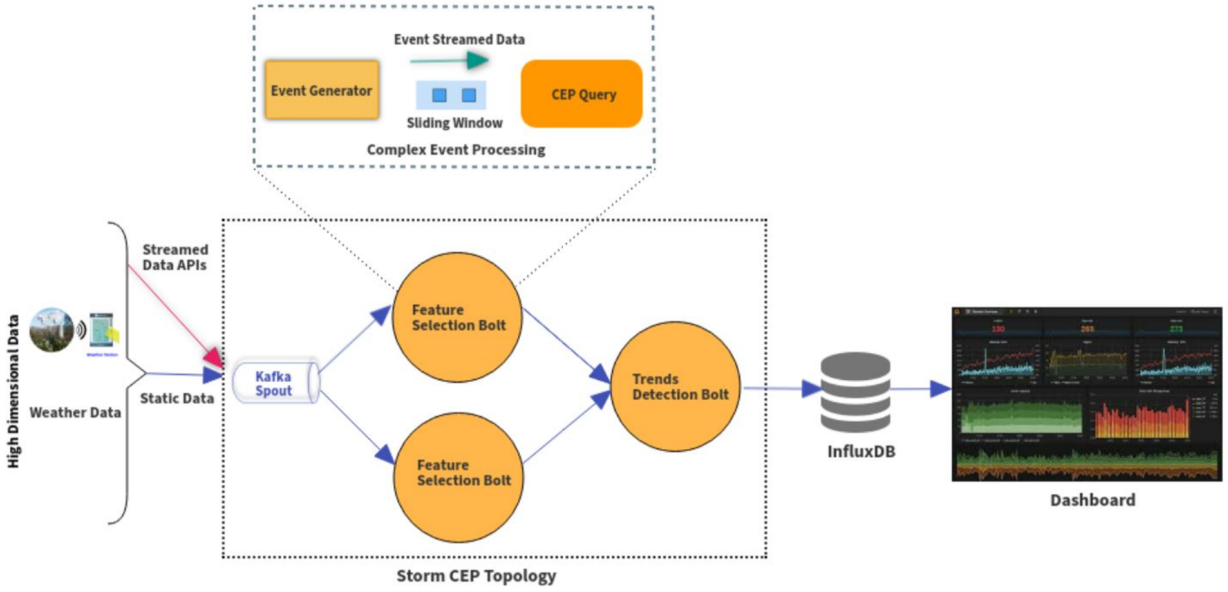


Figure 3: System Architecture: Distributed Storm-CEP Topology

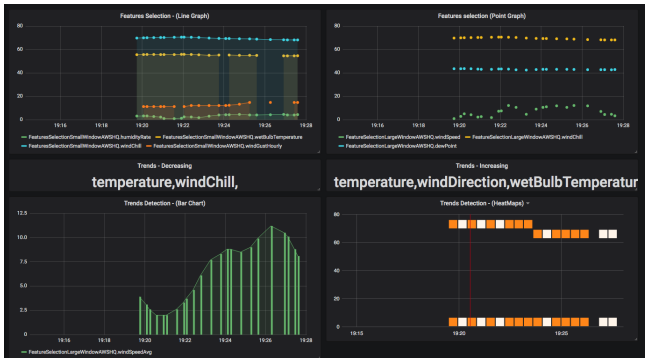


Figure 4: Visualization Dasboard

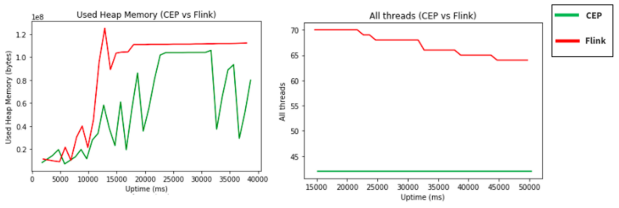


Figure 5: Performance Plot(Storm-CEP vs Flink)

16 *Intel Xeon X5647* processors of 4 cores clocking at 2.92GHz. This machine has a primary memory of 24GB buildup of *DDR3 800/1066* and runs on *Ubuntu* 14.04.5 that has a *x86_64* architecture. We tested our framework with various datasets which were ingested by connecting our system to various data APIs, generating real-time data streams. Finally, we compared the performance of our system(Storm-Esper Distributed Topology) with that of Apache Flink in terms of CPU and memory utilization. As seen in Figure 5, the heap memory used by the system that uses Apache Flink is more as compared to our proposed Storm-CEP system and the overall execution time of our framework is much less compared to the Flink version of the implementation.

## 5 Conclusion and Future Work

In this paper, we have addressed the problem of visualizing high dimensional data streams. For this, we present a generic system that can perform feature selection on real-time high-dimensional data streams with no prior knowledge about features. We show that by using our system, high dimensional data is being projected to lower dimensional space which in turn helps in better exploration by identifying hidden trends. The system is also scalable and can be deployed to multi-node storm cluster in case of high volume and velocity of real-time streaming data for distributive processing. The experiments that we perform help us to understand the trade-off in the performance in terms of CPU consumption and memory and also the performance in terms of execution speed, time, latency, and throughput of our proposed system. To the best of our knowledge, no other system dynamically detects features from streaming data without any prior knowledge about them. The comparative study of the performances between our system and that of the system build on top of Apache Flink gives interesting observations. It has been concluded that when feature selection is performed on the high-dimensional streaming data, the visualizations improve in terms of the rendering performance and reduced visual clutter. Moreover, when the system is combined with a distributive stream processing framework the latency and throughput of the overall system improved resulting in even better performance of the visual exploration tool.

As future work, we would plan to extend our framework by combining our CEP engine with machine learning routines where the system can learn from the historic time-series data and train itself to predict the trends on any given real-time streaming data. By this way, the system can also be extended for predictions and forecasting of trends in multidimensional time-series data.

## ACKNOWLEDGMENT

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

## REFERENCES

[1] Chapter 10. epl reference: Functions. `http://esper.espertech.com/release-6.0.1/esper-reference/html/functionreference.html`, 2020.

[2] I. Flouris, 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*, 127:217–236, 2017.

[3] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos. Continuous trend-based classification of streaming time series. In *East European Conference on Advances in Databases and Information Systems*, pp. 294–308. Springer, 2005.

[4] A. Margara, G. Cugola, and G. Tamburrelli. Learning from the past: automated rule generation for complex event processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pp. 47–58. ACM, 2014.

[5] D. Metz, S. Karadgi, U. Müller, and M. Grauer. Self-learning monitoring and control of manufacturing processes based on rule induction and event processing. In *4th International Conference on Information, Process, and Knowledge Management (eKNOW 2012)*, pp. 88–92, 2012.

[6] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 407–418. ACM, 2006.