Journal of Computer Science

Original Research Paper

Real Time Anomaly Detection in Massive Data Streams with ELK Stack

[1]Jakup Fondaj and [2]Zirije Hasani

[1]*Faculty of Computer Science and Technologies, South East European University, Tetovo*
[2]*Faculty of Computer Science, University of Prizren "Ukshin Hoti", Prizren, Kosovo*

Article history
Received: 13-01-2019
Revised: 23-05-2019
Accepted: 15-06-2019

Corresponding Author:
Zirije Hasani
Faculty of Computer Science,
University of Prizren "Ukshin Hoti", Prizren, Kosovo
Email: j_fondaj@seeu.edu.mk
zirije.hasani@uni-prizren.com

Abstract: Real time anomaly detection is very popular topic nowadays this because the number of data generated every day is larger and larger. Facing with the phenomena of Big Data is not an easy task. The main aim of this research is to fine appropriate architecture for real-time big data analytic and its main task is to detect anomalies in this real-time data. In this paper we show the implementation of anomaly detection algorithm in real time infrastructure in order to find anomalies as soon as possible. We have proposed architecture for real time anomaly detection by adding some new components and the main part of the infrastructure is Timelion which enable implementation of different algorithms for anomaly detection. The research is focused to develop infrastructure to monitor e-dnevnik (education national system in Macedonia) application server and to detect errors in order to scale up the performance.

Keywords: Real Time, Big Data, Timelion, Infrastructure for Anomaly Detection

Introduction

The usage of internet nowadays is constantly increasing the amount of data. As a result, the need for analyzing this data has recently emerged and we need to face a new phenomenon known as massive data streams. This paper shows the appropriate architecture for real-time massive data stream analytics and its main task is to detect anomalies in real-time data. In our previous work (Hasani, 2014a; 2014b; 2015; 2017a; 2017b; Hasani and Fondaj, 2018) we have analyzed various architectures and their suitability to enable real-time anomaly detection in data streams. In this paper, we present the visualization of an e-dnevnik log by using pipeline infrastructure consisting of Redis, Logstash, Elasticsearch, Kibana and Timelion. The first component is Redis which is used as buffer of log data. Logstash have large number of filters which are used to analyze the data, Elasticsearch is indexing component of this infrastructure and also is used for storing data. Other important components are Timelion and Kibana, the first one is used for visualization and the other one for anomaly detection (Hasani *et al*., 2015).

On (Hasani *et al*., 2015) we have explored and implemented different Kibana filters in order to do post-procesing of SQL queries. The focus of the post-processing was to prepare the log information in adequate format and information extraction. The purpose of this analyze was to monitor the performance of the National system of Education in Macedonia and to alert or prevent possible unwanted activities (Hasani *et al*., 2015).

Our research deals with developing infrastructure for monitoring e-dnevnik (education national system in Macedonia) application server and detecting errors in order to enable its smooth work and scale up the performance. Different existing algorithms are implemented in the proposed infrastructure, so our further work will be the implementation of our proposed real-time anomaly detection algorithm for streaming data in this planed infrastructure.

Related Work

Many advanced and highly effective anomaly detection methods exist that run-in batch mode, where the data is collected and processed after the occurrence. However, identifying anomalies long after they happened isn't our main goal. On the contrary, real-time data processing, requests continual input, time-critical manner processing and instant output (e.g., alarm) if anomaly happened. We can model the normal behavior of the data stream instant of searching for unknown anomalies and then we can compare with the observed one (Hasani *et al*., 2018). Consequently, predicting the values of a stream one-time step ahead is used, the deviation between the predicted values and the observed values are measured and a decision mechanism, if an observed value exceeds normal behavior, is established (Hasani *et al*., 2018). Yet other questions arise. The real-time streams are infinite, can have a high rate of data's appearance in time unite (high volume, high velocity) and can evolve over time (Hasani *et al*., 2018). Thus, the development of the model of normal behavior must adapt to these challenges to maintain detection accuracy: be iterative, use only a part of the stream (even before it is permanently stored) and be implemented as a positive feedback in the learning process (e.g., detected anomalies labeling in the supervised process). Due to the need of the real-time detection process, detection algorithms have to be robust, with low processing time (low complexity), even at the cost of the accuracy. Currently, the most intensively developed anomaly detection methods that consider underlined challenges are based on machine learning, neural networks, predictive and statistical time series forecasting models (Hasani *et al*., 2018).

These new features of data and processing requirements in real time, make it inappropriate or impossible to use traditional architectures (infrastructures, parts of the technology solutions and tools with a non-instantaneous processing paradigm) and processing-intensive algorithms to detect anomalies. Typically, according to Bailis *et al*., 2015; Hasani *et al*., 2018), when we talk about systems that give processing in real-time or near real-time, we think for architectures that provide collection and analysis of data streams from several million data in milliseconds/sec, at the time of their arrival, eventually without their permanent storage and if necessary, in conjunction with the previously stored data. According to Joao Gama in 2015 users are looking for answers placed over massive data in real-time. Or, more specifically, it is not only the process of storing Exabyte data in the data warehouse, according to Michael Minelli, co-author of Big Data, Big Analytics, said in Mike Barlo, 2013: "It's about the ability to make better decisions and take meaningful actions at the right time. It's about detecting fraud while someone is swiping a credit card or triggering an offer while a shopper is standing on a checkout line or placing an ad on a website while someone is reading a specific article. It's about combining and analyzing data so you can take the right action, at the right time and in the right place."

The processing of massive data streams in real-time includes multiple different stages, each for itself in terms of the requirements stated above. Also, the possible need for scaling, human intervention, privacy, automatic actions, etc., is a challenge. Consequently, inability to adapt classic architectures and technological solutions, such as Hadoop and similar platforms, which are more convenient for batch analyze (Hasani *et al*., 2018) arise. Requested architecture should provide flexibility in the implementation of the required sequence stages (pipelines): acceptance/write data stream, extraction/cleaning/annotation of data, integration/aggregation/representation, questionnaires/analysis/modeling and interpretation.

The analysis besides its SQL analytic, according to Stonebraker *et al*. (2015; Hasani *et al*., 2018), typically includes machine learning, clustering, predictive modeling, regression estimation, detection of anomalies, etc., which, according to the author, matrix organizing of data compared to the organization in rows or columns gives better performance. Appropriate technology solutions need to give parallel processing flows, scalability, resistant to anomalies and very low latency when giving statements.

Implementation of Infrastructure for Streaming Outlier Detection in Massive Data Streams with ELK Stack

It this chapter the main components and their role in the infrastructure (Redis, Timelion, Logstash, Elasticsearch and Kibana) are explained and filters for pre-processing of SQL log data are shown.

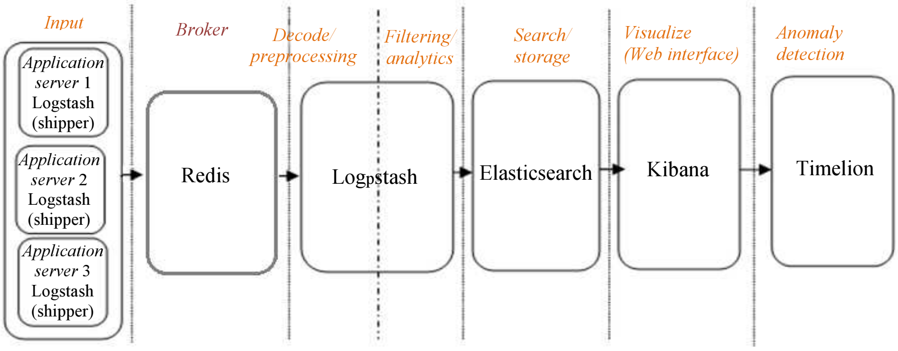

Input
Broker
Decode/ preprocessing
Filtering/ analytics
Search/ storage
Visualize (Web interface)
Anomaly detection
Application server 1 Logstash (shipper)
Application server 2 Logstash (shipper)
Application server 3 Logstash (shipper)
Redis
Logpstash
Elasticsearch
Kibana
Timelion

Fig. 1: Infrastructure for real-time anomaly detection in Massive data streams

Infrastructure for Outlier Detection in Real-Time Massive data streams

To develop the infrastructure, we have done a broad research and based on them we start (Hasani and Fondaj, 2018) with the solution proposed by Kiyoto (2018). The infrastructure is very flexible, it allows us to add and remove infrastructure components very easy. Our proposed architecture is shown in Fig. 1. This architectural design (Hasani *et al*., 2015) is composed from different phases: input (collects and manages events and logs), buffering, decode/pre-process (extract structured data into variables, parse), filter (modify, extract information) and output (ship the data for storage, index, search, visualize and anomaly detection). More details about this infrastructure you can find one our publication (Hasani *et al*., 2015).

The proposed architecture is working in hierarchical form where we have two phases, in the first phase we make data cleaning, filtering and visualization of data and in the second phase is done anomaly detection with Timelion and result visualization from executed anomaly detection algorithms.

The main aim of the infrastructure is to achieve flexibility (Hasani *et al*., 2015), which can be achieved by possibility of adding different new components as Hadoop, Graphite, Cassandra, etc. Generally, in most cases when we run the Logstash server there will be two broad classes of Logstash host (Hasani, 2015):

- The first one is the host which runs the Logstash agent as an event "shipper" that sends application, service and host logs to a central Logstash server
- The second one is central Logstash host which runs a combination of components of this architecture for pre-processing and filtering of events

Broker (usually Redis (2018) is temporary buffer for logs. This is important to collect information when the number of requests is raising unexpected (Hasani *et al*., 2015).

Main component of the infrastructure is Logstash which have three-phase pipeline: input, filtering and output, it has different plugins in every phase enabling extracting information from log data (Hasani *et al*., 2015).

Elasticsearch/Kibana

Elasticsearch enables efficient indexing and storing of the event logs enabling a full-text search on them (Hasani and Fondaj, 2018). It is an open-source distributed search engine library built on top of Apache Lucene (Timelion, 2018). Elasticsearch (Kiyoto, 2018) allows us to implement, store, index and search functionality, this way it helps us in easier and more efficient computation of various data analytics. Elasticsearch is a NoSQL data store (Hasani and Fondaj, 2018) where data are stored as documents. Although it is mainly used by Java applications, the important thing is that applications need not be written in Java to work with Elasticsearch since it can send and receive data over HTTP in JSON to index, search and manage our Elasticsearch cluster (Hasani and Fondaj, 2018).

The last part is Kibana (Kiyoto, 2018). This is an HTML/JS frontend web interface to Elasticsearch for viewing the log data. The beauty of Kibana is that we can easily search the data with different queries, produce charts, histograms and other visual products.

Timelion

The other main component of this infrastructure is Timelion. Timelion is a time series data visualize that enables us to combine totally independent data sources within a single visualization. The main task of Timelion is to find anomalies in real time. It's driven by a simple expression language we use to retrieve time series data, perform calculations to tease out the answers to complex questions and visualize the results. In Timelion we describe queries, make a different transformation of data, implement statistical methods as well as visualized the data to learn from them (Timelion, 2018).

Both Timelion and Kibana provide visualization of the data, but the difference between them is that Timelion provides us with the opportunity to implement the code of the various algorithms used for detection of anomalies and also the result obtained by the execution of these algorithms can be visualized to find out where the anomalies are.

Processing of SQL Queries with Proposed Infrastructure

We have done (Hasani *et al*., 2014a) processing of SQL queries in batch mode, in this paper we have advanced the processing by implementing it in real time environment. The main aim of SQL query analytic is to get information and detect anomalies in order to prevent system down. More details about filters used for SQL query analytic are given in our published paper (Hasani *et al*., 2014a).

Real Time Analysis of e-Dnevnik Database Log File

This part of the paper show how we have analyzed the log files generated from e-dnevnik (Hasani *et al*., 2014a). This application has a large number of requests during the day, we like to check how many requests have the system for given period of time and to detect if there is an anomaly. In the Fig. 2 below we present two histograms produced by Kibana (Hasani *et al*., 2014a).

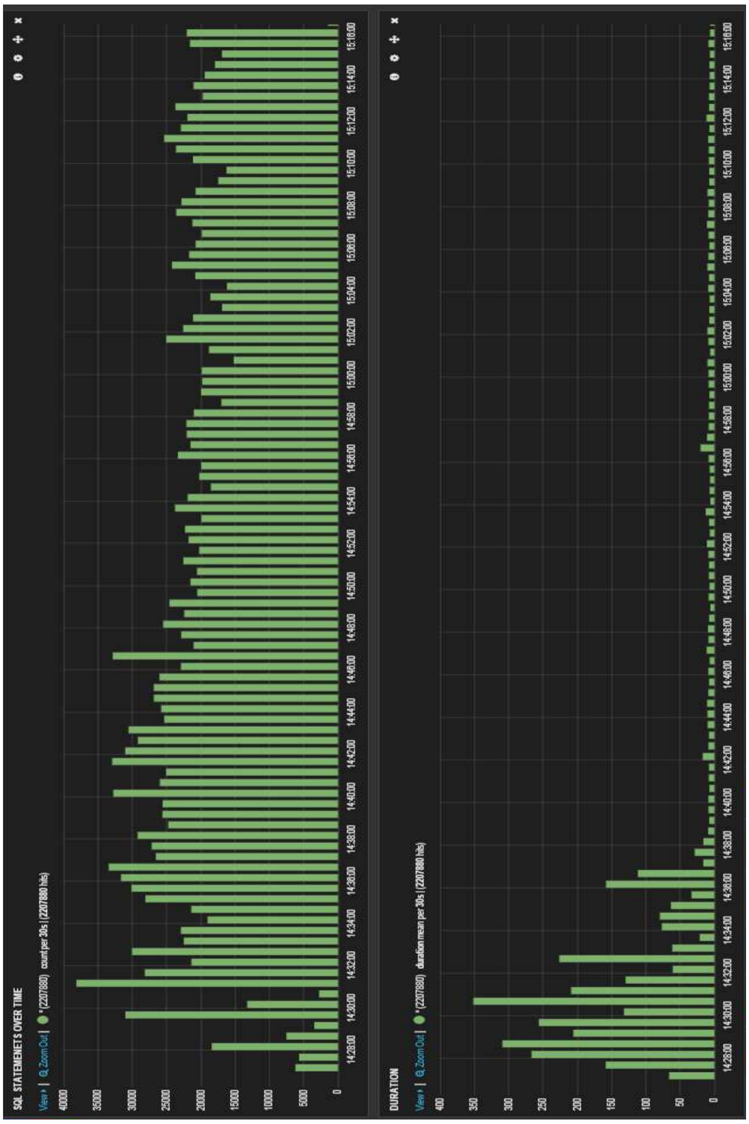


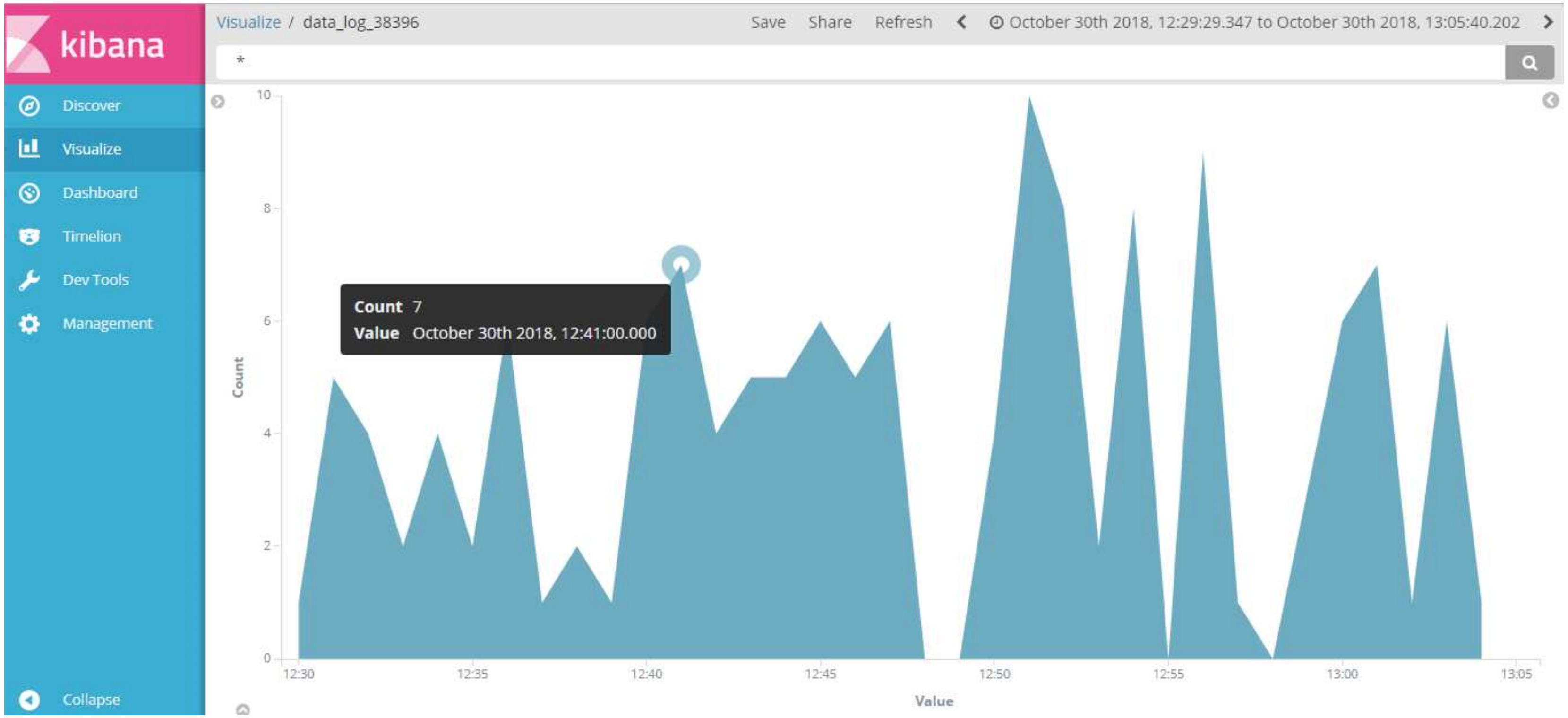
Fig. 2: e-Dnevnik number of hits and duration mean per 30 sec intervals, in selected 14:26-15:16 period of time (Hasani *et al*., 2014a)

The first chart displays the distribution of the number of events in the system, calculated per 30 seconds intervals in the time period from 14:26 untill 15:16, having 2207880 hits all. The second chart shows the calculated mean duration of SQL queries execution time for the same period and intervals. This shows that the mean of the query duration is higher at the specific period of time. The higher mean duration time of SQL queries in this example is the consequence of the Postgres server restart and warming up of Postgres shared buffers (Hasani *et al*., 2014a).

Kibana Visualization of CSV Logs

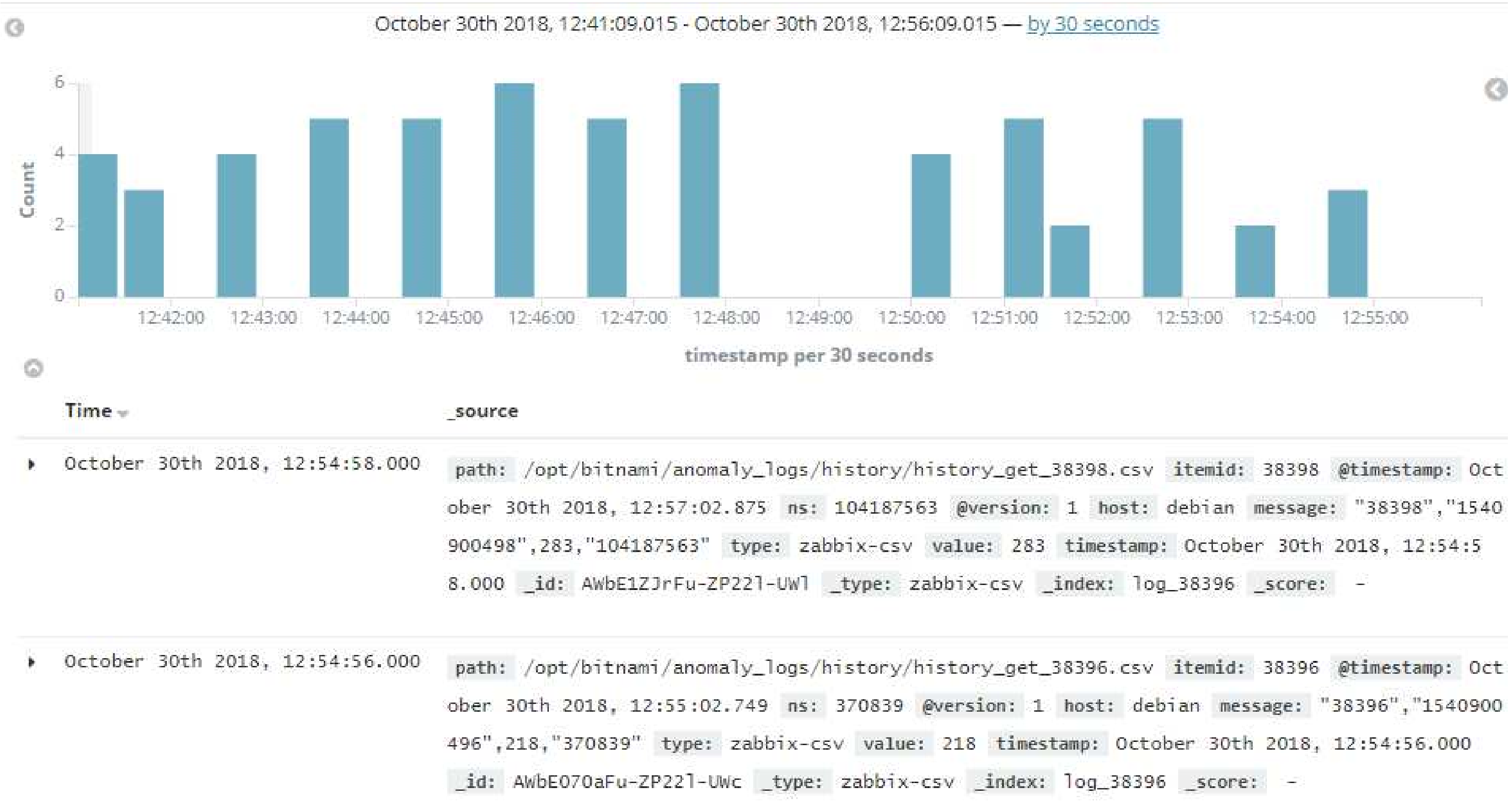
The infrastructure presented in the paper (Hasani and Fondaj, 2018) can be used for anomaly detection in real-time Massive data streams. After configuring Logstash, the data can be visualized in Kibana. With Elasticsearch and Kibana we have the possibility to visualize the log data (Kiyoto, 2018) from our e-dnevnik application servers. In the next figure, we show how the result is visualized by Kibana where we have the possibility to draw our own charts, histograms etc. Kibana offers many functions for data analysis and visualization. Some of their main functions are shown in Fig. 3 and 4, comprising a period of three days. We can see here the different time periods for the number of requests in the chart. We can also see the content of the log file for every event that happens in our application server. The interesting thing about Kibana is that the data can be read and exported in JSON format.

Kibana as a result of visualization produces different attributes which we can use to filter the result.



Count 7
Value October 30th 2018, 12:41:00.000

Fig. 3: Visualization of csv e-dnevnik data in real time by Kibana



```
October 30th 2018, 12:54:58.000   path: /opt/bitnami/anomaly_logs/history/history_get_38398.csv itemid: 38398 @timestamp: Oct
                                  ober 30th 2018, 12:57:02.875 ns: 104187563 @version: 1 host: debian message: "38398","1540
                                  900498",283,"104187563" type: zabbix-csv value: 283 timestamp: October 30th 2018, 12:54:5
                                  8.000 _id: AWbE1ZJrFu-ZP22l-UWl _type: zabbix-csv _index: log_38396 _score: -

October 30th 2018, 12:54:56.000   path: /opt/bitnami/anomaly_logs/history/history_get_38396.csv itemid: 38396 @timestamp: Oct
                                  ober 30th 2018, 12:55:02.749 ns: 370839 @version: 1 host: debian message: "38396","1540900
                                  496",218,"370839" type: zabbix-csv value: 218 timestamp: October 30th 2018, 12:54:56.000
                                  _id: AWbE07OaFu-ZP22l-UWc _type: zabbix-csv _index: log_38396 _score: -
```

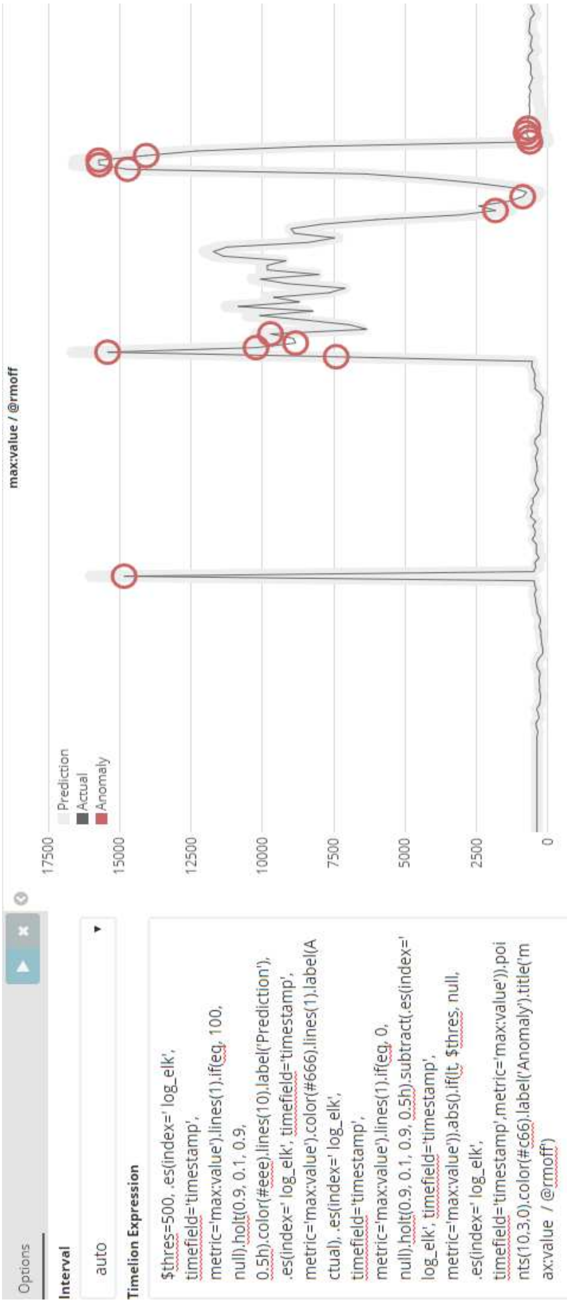
Fig. 4: CSV messages in Kibana

Anomaly Detection with Timelion

An anomaly in our case is considered to happen when the number of requests is increased during a period of time from 19:00 pm to 06:00 am, as non-working hours, or in non-working days, Saturdays and Sundays. The infrastructure that is used receives the CSV log data as an input from the application servers of e-dnevnik.

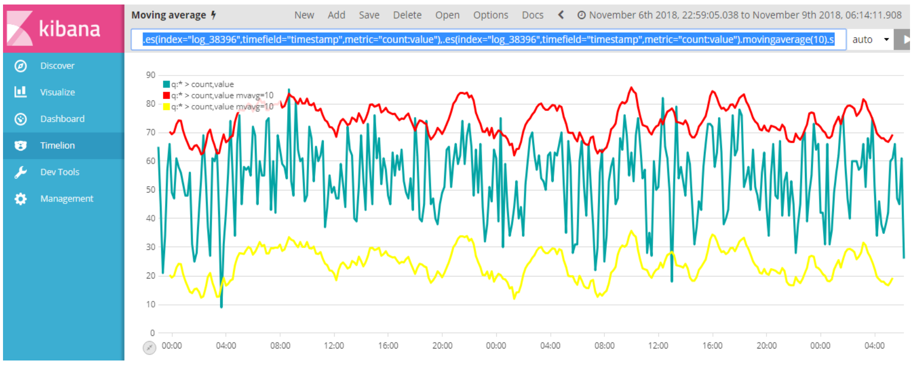
As part of this infrastructure, Timelion, enables programming and implementation of user defined anomaly detection algorithms, as example algorithms we analyzed in (Hasani, 2017). There are some algorithms used for anomaly detection and we explore some of them and implement in real time environment in our proposed infrastructure.

The following figure shows the results of the implementation of Triple Exponential Smoothing, the Holt-Winters method (Kibana, 2018). Timelion have user interface where we can write the commands as is shown in Fig. 5 and by executing them in real time we can see the result in our case detection of anomalies.



```
$thres=500, .es(index=' log_elk',
timefield='timestamp',
metric='max:value').lines(1).if(eq, 100,
null).holt(0.9, 0.1, 0.9,
0.5h).color(#eee).lines(10).label('Prediction'),
.es(index=' log_elk', timefield='timestamp',
metric='max:value').color(#666).lines(1).label(A
ctual), .es(index=' log_elk',
timefield='timestamp',
metric='max:value').lines(1).if(eq, 0,
null).holt(0.9, 0.1, 0.9, 0.5h).subtract(.es(index='
log_elk', timefield='timestamp',
metric='max:value')).abs().if(lt, $thres, null,
.es(index=' log_elk',
timefield='timestamp',metric='max:value')).poi
nts(10,3,0).color(#c66).label('Anomaly').title('m
ax:value  / @rmoff')
```

Fig. 5: Anomaly detection in timelion

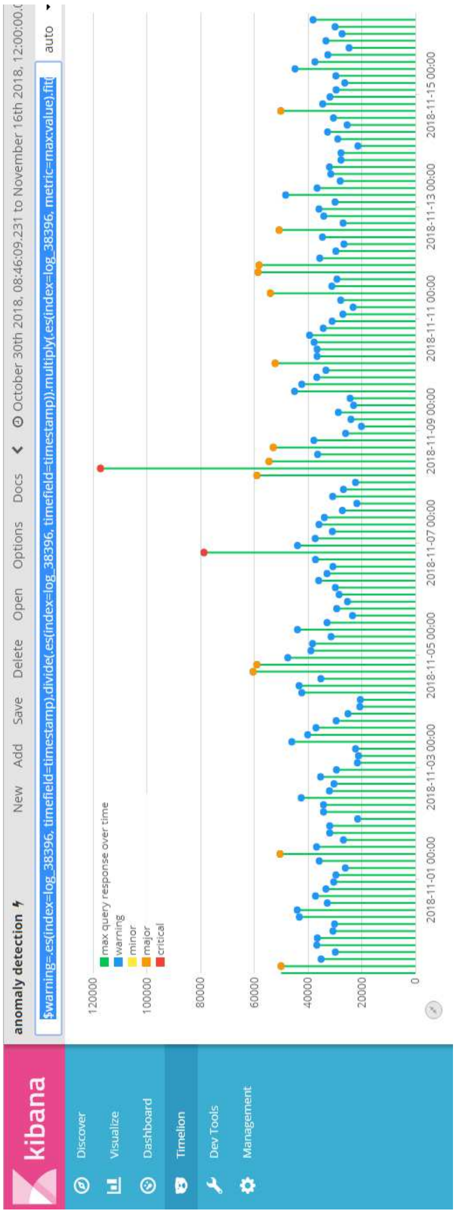


Moving average

November 6th 2018, 22:59:05.038 to November 9th 2018, 06:14:11.908

```
.es(index="log_38396",timefield="timestamp",metric="count:value"),.es(index="log_38396",timefield="timestamp",metric="count:value").movingaverage(10).s
```

Fig. 6: Moving average in timelion



anomaly detection

October 30th 2018, 08:46:09.231 to November 16th 2018, 12:00:00.(

```
$warning=.es(index=log_38396, timefield=timestamp).divide(.es(index=log_38396, timefield=timestamp)).multiply(.es(index=log_38396, metric=max:value).fit(
```

Fig. 7: Anomaly detection based on different conditions

From these results, we can see the anomalies pointed with red circles. These anomalies are human annotated and in the above diagram we can prove that the algorithm works well.

Moving Average in Timelion. One of the most used statistical methods for anomaly detection is Moving Average. In this example we have implemented this method in real time environment. There are defined upper and lower bound in order to check for anomalies. If a point of data is outside this bound is considered anomaly.

Anomaly detection based on different conditions. This algorithm divides data with AVG multiply with 100 and then give 4 conditions for anomalies which may be just warnings then, minor anomalies, major and critical. This mean this method categorize anomalies into four groups, the result is shown in Fig. 6. The Fig. 7 shows the result we get from this algorithm and the visualization of the result with Timelion.

The idea is all this method which we use to compare them and to decide which is the best methods based on our needs and also to propose our algorithm. Based on the figure we get; we go further with the analysis and find the reason the anomalies happened at a specific time period. The number of anomalies depends on the threshold we define, in our case the smaller the threshold the larger the number of anomalies appears.

Conclusion

Big Data analytic is main topic nowadays and is not easy task because the challenge came from the fact that the data are analyzed in real time. In this paper, we propose infrastructure which enable real time Big Data analytic where e-dnevnik log data are analyzed.

The infrastructure proposed is composed from different open source tools as: Redis, Logstash, Elasticsearch, Kibana and Timelion. The infrastructure design is based on the pipeline event processing, divided in phases: input (collects and manages events and logs), buffering, decode/pre-process (extract structured data into variables, parse), filter (modify, extract information) and output (ship the data for storage, index, search, visualize and anomaly detection). Proposed architecture is capable to scale up/out depending on the input stream size and rate, by running one or more of its components as separate threads/servers. Flexibility is achieved by possibility of adding various further components as Hadoop, Cassandra, statistical or graphical tools like Statsd, Graphite, or deploying extension of functionalities in each phase by using own plugins (Hasani *et al*., 2015).

We illustrate the SQL queries database transaction logs analytics with implementation of the filters that produce various statistics enabling detections of anomalies in query performance on an operational level (Hasani *et al*., 2015). This means that we are able to detect performance degradation of SQL queries in real time and alert adequately in order to remove the possible causes. In the same time in real time we do the pre-processing of the logs in order to reduce the amount of content of SQL queries that are necessary to be saved for further analyze.

The testing is done in real-time data that comes from the e-dnevnik application. This tool enables us to implement our evaluated algorithms (Hasani, 2017) and visualize the results. Visualization of the real-time anomaly detected is an important part of the infrastructure as a powerful tool for online monitoring the work of the system. The main contribution of the paper is that propose real time infrastructure which enable implementation of different algorithms for anomaly detection in real time streaming data and also preprocessing of data is done.

Author’s Contributions

Author 1 do the research and write the paper and author 2 do the experiments and revise the paper.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Bailis, P., J.M. Hellerstein and M. Stonebraker, 2015. Readings in Database Systems. 5th Edn., RedBook, USA.

Hasani, Z. and J. Fondaj, 2018. Improvement of implemented infrastructure for streaming outlier detection in big data with ELK stack. Proceedings of the 6th World Conference on Information Systems and Technologies, (IST’ 18), Springer, Italy, pp: 869-877.

Hasani, Z., 2015. Performance comparison throw running job in Hadoop by defining the number of maps and reduces. Proceedings of the 12th International Conference on Informatics and Information Technologies (IIT’ 15), Bitola, Macedonia, pp: 49-53.

Hasani, Z., 2017. Robust anomaly detection algorithms for real-time big data: Comparison of algorithms. Proceedings of the 6th Mediterranean Conference on Embedded Computing, (CEC’ 17), IEEE, Montenegro, pp: 1-6.

Hasani, Z., 2017a. Implementation of Infrastructure for Streaming Outlier Detection in Big Data. In: Recent Advances in Information Systems and Technologies, Rocha, Á., A. Correia, H. Adeli, L. Reis and S. Costanzo (Eds.), Springer, Cham, pp: 869-877.

Hasani, Z., 2017b. Infrastructure with R package for anomaly detection in real time big log data. Pressacademia, 5: 181-189.

Hasani, Z., B. Jakimovski, G. Velinov and M. Kon-Popovska, 2018. An adaptive anomaly detection algorithm for periodic data streams. Proceedings of the 19th International Conference on Intelligent Data Engineering and Automated Learning, Nov. 21-23, Madrid, Spain, pp: 385-397. DOI: 10.1007/978-3-030-03493-1_41

Hasani, Z., B. Jakimovski, M. Kon-Popovska and G. Velinov, 2015. Real time analytic of SQL queries based on log analytic. ICT Innovation, Ohrid Macedonia.

Hasani, Z., M. Kon-Popovska and G. Velinov, 2014a. Survey of technologies for real time big data streams analytic. Proceedings of the 11th International Conference on Informatics and Information Technologies, (IIT' 14), Bitola, Macedonia, pp: 321-326.

Hasani, Z., M. Kon-Popovska and G. Velinov, 2014b. Lambda architecture for real time big data analytic. Proceedings of the Web ICT Innovations, (WII' 14), Ohrid, Macedonia, pp: 133-143.

Kibana, T., 2018. Anomaly Detection. https://rmoff.net/2017/01/18/kibana-timelion-anomaly-detection/

Kiyoto, T., 2018. Elasticsearch, fluentd and kibana. Open Source Log Search and Visualization. https://www.digitalocean.com/community/tutorials/elasticsearch-fluentd-and-kibana-open-source-log-search-and-visualization

Timelion, 2018. https://www.elastic.co/guide/en/kibana/current/timelion.html

Joao Gama, 2015. Challenges in Learning from Streaming Data. Bogdanova. In: ICT Innovations 2014, Advances in Intelligent Systems and Computing, Ana Madevska, Gjorgjevikj, Dejan (Eds.), Springer, Switzerland, pp: 1-5.

Mike Barlo, 2013. Real-Time Big Data Analytic: Emerging Architecture. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA.