Building an IoT Data Hub with Elasticsearch, Logstash and Kibana

Conference Paper · August 2017

DOI: 10.1109/FiCloudW.2017.101

1 author:

Marcin Bajer
ABB
27 PUBLICATIONS **65** CITATIONS

SEE PROFILE

Building an IoT Data Hub with Elasticsearch, Logstash and Kibana

Marcin Bajer
ABB Corporate Research Center
Starowiślna 13a
31-038 Kraków, Poland
Email: marcin.bajer@pl.abb.com

***Abstract*—The goal of this publication is to describe the implementation of an Elastisearch, Logstash and Kibana (ELK) stack to process IoT data. Although, those tools were designed to be used mainly for handling large number of log data, they can be applied also to store, search and visualise other type of information - including IoT data. In order to show practical implementation of the idea, selected devices installed in the building of ABB Corporate Research in Krakow has been used. In prepared system, real data generated by various subsystems in this building has been integrated into one Elasticsearch based solution for further processing. Later on, this data will be used to develop data analytics to extract and visualize meaningful insights about building operations. In addition, selected data is sent to the Azure cloud to use its abilities in big data processing and machine learning.**

I. Introduction

The Internet has completely transformed people's life. It touches on nearly every aspect of modern society. It has affected the way people communicate to the extend for many it is now the preferred medium of everyday communication. It has changed also the way how information is being stored and searched. Webpages, ebooks, blogs and wikis are gradually replacing printed newspapers and books. By the end of 2016 more than 47% of world population were Internet users which made the Internet the most popular source of information [1]. We have seen a revolution triggered by the fixed Internet connection to PCs and laptops. Currently, we are witnessing the emergance of mobile Internet in smart phones and tablets. In fact, ubiquitous communication and connectivity is no longer a challenging task [2]. The focus is now shifted toward ensuring seamless integration of people and devices to translate the physical realm into its virtual representation [3].

The Internet is evolving from method of person-to-person communication and person-to-machine interaction into mechanism for direct and automatic machine-to-machine (M2M) cooperation. In fact, M2M communication concept is older than the Internet. Decreasing cost of hardware, new high performance CPUs and recent technology advances (i.e. low power wireless connectivity, 4G mobile communication) allowing ubiquitous connectivity and big data analytics accelerated implementation of the Internet of Things (IoT) concept. Nowadays, data amassed from IoT devices can be easily analysed in the cloud to optimize products or offer new services and operations. This opens a whole set of new business

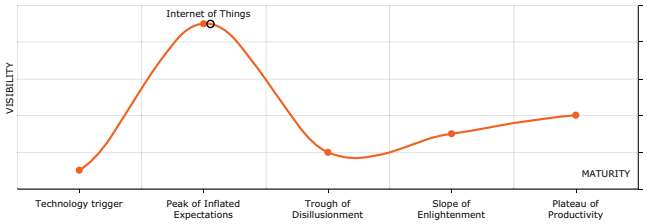

Fig. 1. Garner Hype Cycle for Emerging Technologies 2015 [7]

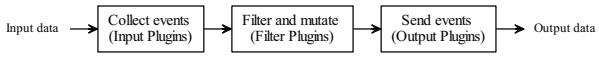

Fig. 2. Logstash processing flow

opportunities [4]. In many cases, hardware and software are no longer the issue, because it is off-the-shelf available. The same hardware and software can be used in many different solutions, the distinct feature is the system logic.

There are great expectations in IoT, it is thought of as the most technology in the following decades. McKinsey estimates that the IoT will have a potential economic impact of $11 trillion per year by 2025 - which stands for about 11% of the global GDP by that time [5]. To predict how IoT will evolve over time we can refer to Garner Hype Cycle for Emerging Technologies (see Figure 1) which provides a graphic representation of the maturity and adoption of new technologies and applications. IoT is now almost at the peak of expectations. There is a lot of publicity in this topic, researchers and companies involved in are showing first success stories [6]. On the other hand it is clear that there are still challenges that need to be worked out. Privacy, security and combining heterogeneous data sources are only few of them. The Hype Cycle shows there is still long way to go before this technology will gain full adaptation and there is much room for potential failure.

II. Elastic stack

ELK stack is a set of three tools - **E**lasticsearch, **L**ogstash and **K**ibana. Although, Logstash and Elasticsearch can work separately, the three products are designed to be used as an integrated solution, currently referred to as the Elastic Stack. A fourth product called Beats was subsequently added to

978-1-5386-3281-9/17 $31.00 © 2017 IEEE
DOI 10.1109/FiCloudW.2017.101

the stack as a lightweight shipper that sends data from edge devices to Logstash. The Elastic Stack can be deployed on-premises or used as Software as a Service (SaaS) provided by external company in the cloud. The stack is currently maintained and actively supported, under open source license [8], by the company called Elastic. There is also a set of community plugins developed by volunteers.

III. Logstash

Logstash is a plugin-based event forwarder with many features. It ingests data from multiple sources simultaneously, transforms it, and then ships it elsewhere. Each event is processed in a three-stage pipeline: inputs → filters → outputs.

First, there is an input plugin which allows to handle a specific input stream. Logstash supports a variety of input types and allows to capture events from all common data sources including CSV file, TCP/UDP socket, HTTP API endpoint, Elasticsearch and many others.

Next, the processed event can be mutated to alter it, remove dispensable data or add additional information. Logstash provides plugins for many different data operations. Filters are often applied conditionally and can perform complex operations. For example, with Grok filter it is possible to extract data from a string field containing text with known pattern (Listing 1) and with Ruby filter it is possible to execute custom Ruby code for event processing.

Finally, when outputting data, Logstash supports a huge range of destination types. Usually, all events are sent to Elasticsearch, but Logstash can be used also independently to save data to a CSV file, an SQL database, data analytics algorithm (i.e. Azure Machine Learning) or just show it to the console for debugging purposes. For each data source (i.e., single type of data sensor) it is required to prepare dedicated configuration file for Logstash (Listing 2). Sections of this file refer to the particular stages of event processing pipeline. Operations inside each section are applied in the order of declaration, but sections processing is done in strict order (Figure 2). It is not possible to specify additional filter after output section, in case it is required to create two different output events out of single input event you need to duplicate the event, add distinguishing mark to one of the event copies and process it conditionally.

```
filter {
 if [syslog_program] == "dnsmasq" {
  grok {
   match => { "syslog_message" => "query\[%{DATA:
       query_type}\] %{USERNAME:search_url} from %{
       IP:client_ip}" }
  }
 }
}
```

Listing 1. Grok filter for parsing log data from dnsmasq application

In addition, to specifying how data enter and leave Logstash, it is required to specify the format of this data. In most cases plain text or JSON is used as the codec, but in some cases more elaborate parsing might be needed (i.e. gzip encoded content, base64 encoding or collectd output).

```
input {
  exec {
    command => "curl --silent -L 'http://api.
        openweathermap.org/data/2.5/weather?id=
        YOUR_ID&APPID=YOUR_APPID&units=metric'"
    codec => "json"
    interval => 600
  }
}
filter {
  mutate {
    remove_field => [ "@version","command","host", "
        cod","id","base","coord","sys","dt" ]
  }
  split { field => "weather" }
}
output {
  elasticsearch {
    hosts => [ "10.3.100.41:9200" ]
    index => "weather-%{+YYYY.MM.dd}"
  }
}
```

Listing 2. Logstash configuration for reading weather data

To split data into multiple Elasticsearch indexes Logstash Elasticsearch output plugin accepts specification of the index pattern. An index pattern is a string with optional wildcards that can match multiple Elasticsearch indices. This allows to automatically redirect document to proper index based on its timestamp or content (i.e. split indices per day, per event type or source).

IV. Elasticsearch

Elasticsearch was build on top of Apache Lucene, is an open source, distributable, and highly scalable search engine which was designed to have optimal performance. It implements inverted indices with finite state transducers for full-text querying, BKD trees [9] for storing numeric and geo data, and a column store for analytics [10]. It has some proven large scale deployments, such as GitHub [11] or Stack Overflow [12] to name just two.

Elasticsearch has a comprehensive REST-based API which allows to monitor and control all aspect of a cluster configuration. It provides endpoints to perform **c**reate, **r**etrieve, **u**pdate and **d**elete (CRUD) operations on stored data over HTTP API calls. To some extend Elasticsearch can be used alike any other NoSQL data store [13]. Complex search requests, including those that involve faceting and statistical operations, can be realized in JSON-based Query DSL (**D**omain **S**pecific **L**anguage). Elastic provides wrappers for the API in popular programming languages such as Java, Python, .NET, and Groovy. In addition, many other languages are supported by the community.

Unfortunately, updating a single document from Elasticsearch index is expensive in terms of performance. If execution of large number of update operations is needed it is better to perform them as one single Bulk API operation. In case data retention mechanism is required, it is recommended to split data into multiple indices base on timestamp. Delete operation is performed on whole index. Depending of number of data, indices can be created on monthly, weekly, daily or even hourly basis.

A. Mapping

Elasticsearch is schema-less database, it is sufficient to send JSON object as the input and it creates *document* with appropriate mapping for each JSON field automatically, with no performance overhead. It is also possible to define mapping implicitly, but it must be done before adding first value of that type, because it is not allowed to alter existing field mappings.

Elasticsearch provides dedicated API for mapping configuration. The default type mapping has reasonable default values, but when you want to change default mapping or customize indexing (storing, ignoring, completion and so on) you need to provide a own mapping definition [14]. By default dynamic mapping is enabled. To prevent indexing unneeded data it is possible to disable dynamic mapping for selected parts of document - in this cases inner fields without static mapping defined will be skipped.

For integers, it is the best to select the smallest possible type applicable in particular use-case. This will not reduce index size, but help indexing and searching be more efficient [15]. It is also possible to map floating point values into integers using *scaling factor*.

In some cases, Elasticsearch can not deduct automatically correct mapping - for example string field can be indexed as *text* field for full-text search, and as a *keyword* field for sorting or aggregations. By default, system will index this field in both ways. If you know the purpose of such field you can prevent overhead and select in advance how it will be indexed.

It is also possible floating point fields will be wrongly mapped as integers because the data source sends value without floating part when it is zero (i.e. 123 instead of 123.0). In such case mapping for this field need to be predefined, because if field will be mapped as integer and system will reject whole record if at least one of its fields has wrong type.

```
PUT 'http://ELASTICSEARCH_IP:9200/_template/weather'
{
   "template":"weather-*",
   "settings":{"number_of_replicas":4},
   "mappings":{
      "logs":{
         "properties":{
            "temp":{"type":"byte"},
            "icon":{"type":"keyword"},
            "wind":{
               "properties":{
                  "deg":{"type":"short"},
                  "speed":{
                     "type":"scaled_float",
                     "scaling_factor":10
                  }
               }
            }
         }
      }
   }
}
```

Listing 3. Index template for *weather-** indices

The mapping can be created together with index, but this is a bit problematic in case Logstash create new indexes base on selected pattern (i.e. one index per day). In such case it is possible to define index templates which will be automatically applied when new indices are created. Index template contains index settings, type mappings and pattern base on which system decides if the template should be applied to the new index (Listing 3).

V. Kibana

Kibana was designed as a visualization platform for Elasticsearch. It provides web-based interface for search, view and analyse data stored in Elasticsearch cluster. The main view of Kibana is divided into 4 main components - *Discover*, *Visualize*, *Dashboards* and *Management*.

The *Management* section is where all Kibana internals can be configured. It is a place where index patterns must be set. If index patterns will contain time-base events it is also needed to specify the the timestamp field base on which Kibana will sort and filter data. Kibana, base on set of indices which fulfil selected index pattern, shows the list of index fields together with their type and properties. In addition, it is possible to modify field formatters to alter how the field value is shown in Kibana GUI.

Discover allows to interactively browse and analyse pure data entries. Elasticsearch documents are categorized base on Index patterns defined in *Mangement* section. They can be easily searched and filtered by time or by document properties using Apache Lucene query syntax. One can also see the number of documents that match the search query or get field value statistics.

Due to the plugin based architecture, Kibana can be easy extended to suits particular needs. *Visualize* section provides possibility to visualize data with one of provided visualization plugins. Information can be shown in form of tables, charts, maps, histograms and many others. Large volume can be easily shown in form of pie chart, bar chart, line or scatter plot. Unfortunately, functionality to export raw data from Elasticsearch is missing. To fix this problem one can use ES2CSV tool [16], which allows to export Elasticsearch data in CSV form

Dashboard allows to combine multiple saved *Discover* and *Visualize* results into one view. It is possible to arrange and resize the dashboard elements as needed. Having some experience it is possible to create very sophisticated and colourful dashboards, directly from Kiban GUI. Dashboards can be easily saved, shared or embedded into other web page.

The main drawback of Kibana is the fact, out of the box, it has no user management. The simplest solution to restrict access to selected users is to use Nginx proxy forwarder [17]. If more elaborate access control is required, it is possible to purchase commercial support from Elastic to get a license for *X-Pack* - this will no only allow to configure sofisticated security options, but also enable reporting and alerting features. It is also possible to use Searchguard [18] - this Elasticsearch plugin offers encryption, authentication, and authorization. The basic security features are for free. Enterprise features are licensed for commercial use and free of charge for personal and non-commercial projects. Another

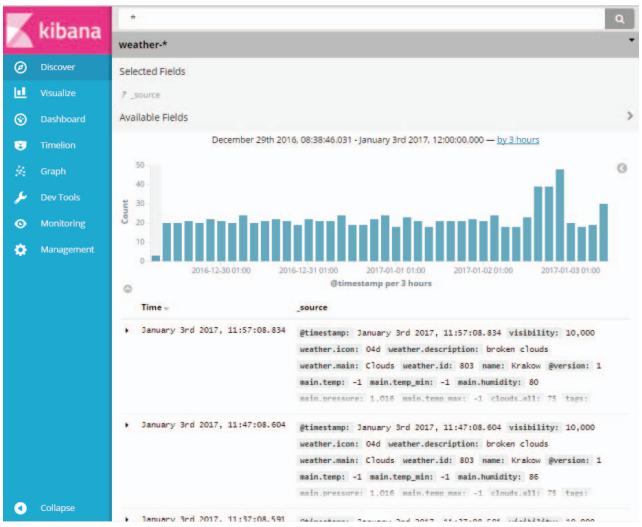

Fig. 3. WWW interface of Kibana

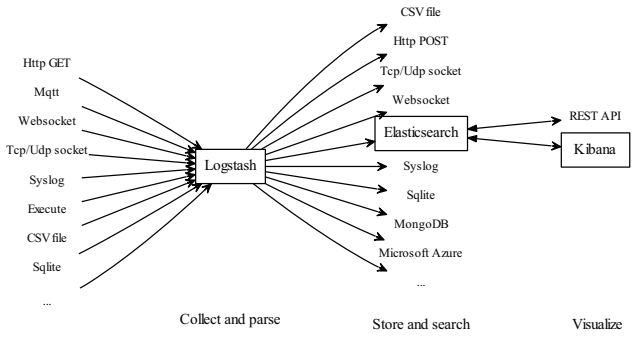

Fig. 4. ELK stack

alternative is ReadonlyREST plugin [19], which offers quite similar set of features under the GPLv3 license.

VI. Scaling

Scaling mechanism is very important in case of developing IoT solutions. Very often it is not known in advance how many devices system will end up with and what will be the computation power needed. If the business idea is a good one, system can easily grown to thousands of nodes spread around the world. If not, there is no reason for investing time and money to solve problems which do not exists in smaller systems.

Elasticsearch supports two types of scaling:

Vertical scaling The simplest method of increasing capacity of search engine is to add more hardware resources (CPU cores, RAM or RAID disk). Of course, due to the technology limitations and cost factor this process is not infinite.

Horizontal scaling Alternative method is to add more machines to the cluster. Elasticsearch will automatically split records to be searched between available Elasticsearch nodes and merge the search results from all machines.

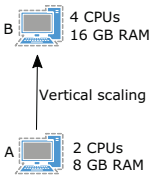

Fig. 5. Vertical scaling

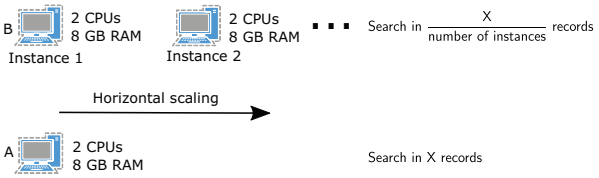

Fig. 6. Horizontal scaling

The data in the Elasticsearch is stored in *indices* which are similar to SQL tables. Each index is divided into one or more smaller pieces called *shards*. Elasticsearch manages automatically primary shards containing original information and replica shards which are copies of this information spread around cluster nodes to increase performance and durability. Registering a new node to a cluster is a matter of installing Elasticsearch on a machine and editing a configuration file. Elasticsearch automatically takes care of spreading data around and splitting out requests over multiple servers.

VII. Real life use case example

To check the feasibility of implementing Elasticsearch engine for processing data form IoT devices it was decided to use various devices installed in building of Corporate Research Center in Krakow, Poland. The Elastic Stack was deployed on virtual machine with 16GB RAM, SSD disk and dual Xenon 2.4GHz core running Ubuntu 16.04. The standard configuration of system has been altered to increase Java Virtual Machine Heap to 4GB. Although, current configuration is sufficient to handle all incoming IoT data, it is planned to add second Elasticsearch instance on new virtual machine to ensure data backup and speed-up data search.

The overall system has been described in Figure 7. It consists of multiple isolated subsystems. Most of them are not designed to be integrated with any communication system for smart buildings (KNX, BACnet, LonWorks). Usually, they offer Modbus TCP connection for accessing data or REST API interface for reading data via regular HTTP requests. In the end, approximately 600 different signals were captured with frequency of 10s. Kibana is used to visualize building operations and show it on multiple TV screens installed throughout the building.

It was very simple to prepare a dedicated mechanism for reading all necessary data using Logstash. Although, there are is no available Logstash plugin for Modbus TCP it was quite easy to prepare a mechanism for reading Modbus TCP data. Since Node.js offers significant amount of off-the-shelf

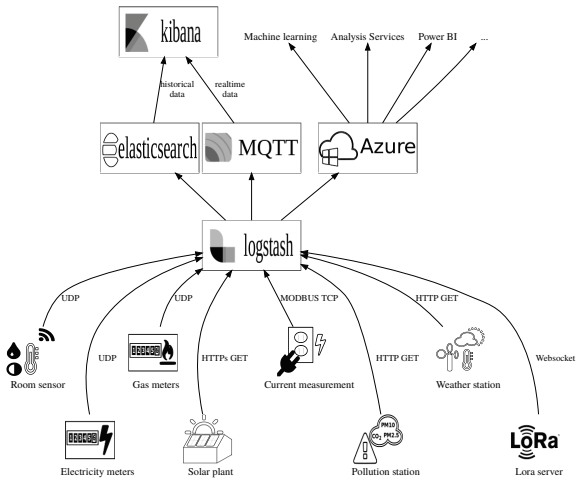

Fig. 7. Implementation of Elasticsearch base system in ABB CRC Krakow

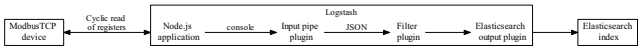

Fig. 8. Reading data from ModbusTCP device

modules for various interfaces, including several for Modbus TCP. It was decided to use one of them for reading data from Modbus devices. To optimize index search, appropriate data mapping was prepared in advance using Elastisearch mapping API. Dedicated Node.js application was prepared to periodically read from the device and print data to console in JSON format. The application is triggered via Logstash Pipe input plugin which also intercept console output, send it through the Logstash processing flow further to Elasticsearch (Figure 8). Of course, one can say it is not the best solution in case of performance (especially RAM usage), but in case of several types of devices this has no impact.

In case of accessing device data via HTTP REST API, Logstash HTTP input plugin was used. It supports basic HTTP access authentication which is usually enough. If custom authorization algorithm is required or additional (complicated) data processing is needed, it is possible to use similar approach as in case of ModbusTCP - dedicated Node.js application and Logstash pipe input plugin.

VIII. Kibana plugin development

Unfortunately, Kibana does not offer public API to develop against it. Each new version introduces changes in API. Since it is necessary that installed plugins must match exactly the version of Kibana itself, plugin developer have to release a new version of the software for each new Kibana release. Nevertheless, Kibana API does not vary a lot between subversion of major release - it is easy to customize plugin to new version of Kibana.

Currently, available set of Kibana visualization plugins allows to easily plot historical data in various ways, as well as to show it in tabular form. In case custom view is

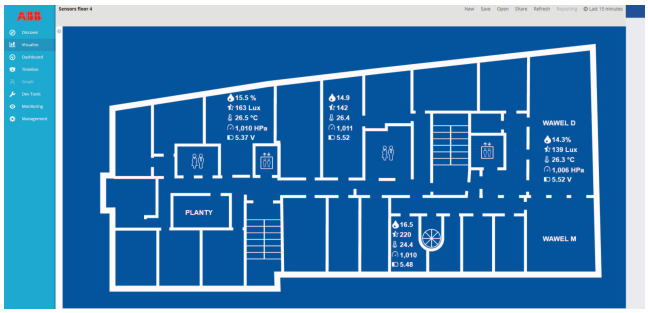

Fig. 9. MQTT visualization plugin

needed, one need to develop own plugin or even separate application embedded into Kibana. Assuming some experience in development of web applications in Node.js and AngularJS it is possible to quickly prepare dedicated plugin. There is a set of dedicated tutorials how to write own Kibana plugin [20], [21], [22], but the best source of information is the source code of basic plugins bundled into Kibana source code. In case of presented system, dedicated plugin to show realtime data in visualization was developed (Figure 9). To show current readings from sensors spread throughout the building SVG graphical model was extended with AngularJS tags and directly used as plugin view. The controller uses MQTT client implementation [23] which connect directly to MQTT server [24] via websocket to get live sensor data and inject it to the plugin view (Figure 7).

IX. Azure integration

One of the biggest challenges of analysing data in Elasticsearch — and indeed in any big data set — is the ability to find the information that matters. In many cases, new data sources are deployed to the system without clear definition what to do with this new information flood and what will be the added value. Eventually, even when the data and its accuracy have a high value, one has to be aware that this information might keep its high value for only a very limited time [25]. Collected data might be useful for optimization of long time system performance, but they also may be indicators of a catastrophe about to take place that will have the potential to seriously impact the system - in this case immediate reaction is needed.

Because of the volume, it is required to do data analyses in an automatic way. The idea for further extension of presented system is to combine it with Azure Machine Learning and use cloud against data recorded over a period of time to identify trends and correlations between data/events. In this particular case data analytics can be applied to extract and visualize meaningful insights about building operations.

Exemplary solution for data sending to Azure has been described in Figure 10. To transfer data between Logstash and the cloud it is possible to use one of the three protocols: MQTT, AMQP or HTTPS. Due to the ABB's internal network limitations (firewall blocking ports) the best option was to use HTTP REST API. IoT events are consumed by Azure Event

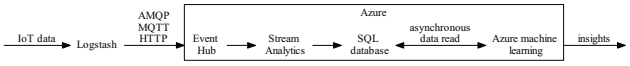

Fig. 10. Exemplary implementation of Azure Machine Learning

Hub and send to Stream Analytics block which is configured to save selected data to SQL database. This database is asynchronously read by machine learning algorithm which provide to user results of performed data analytics.

X. ELK ALTERNATIVES

There are few other equivalent solutions to ELK stack. First of all, there is Splunk, which is well know, comprehensive and feature rich, but unfortunately not open source product. Splunk can be used not only for logs, but also for IoT data. For example, the makers of the Nest thermostat use Splunk to analyze data uploaded from hundreds of thousands of homes, and tune their algorithms for energy performance [26].

Graylog is open source alternative to ELK stack. It is also using the Elasticsearch for data storage and MongoDB as metadata store. Although, Graylog uses its own mechanism for data collection, with some additional work it can be configured to work with Logstash.

As an alternative to Elasticsearch based solutions it is possible to use the Grafana. Grafana is open source web based dashboard tool similar to Kibana. It commonly works with InfluxDB or OpenTSDB databases. Grafana is usually used to monitor IT infrastructure performance metrics, but it can be applied to IoT data as well. Both InfluxDB and OpenTSDB are open-source time-series based databases which can be easily scaled to handing large volume of data. Sensors can directly write to the database via HTTP API. Those databases provides also out-of-the box support for mathematical and statistical functions which can be directly applied on IoT data. Reading from database is also done via HTTP API using SQL-like queries.

Last but not least, there are cloud based solutions like Loggly, Sumo Logic or PaperTrails. They are designed to be used for log processing. Their usage for IoT data analyses is limited, but in some cases it can be a good startup solution to quickly setup system and start logging data.

XI. CONCLUSION

In this paper, a solution for integration of data from multiple subsystems into one database is presented. The test has shown that this solution is working and provide significant amount of flexibility in configuration. It has already proven its usage for data analyses. Since Elastic stack provides not only versatile data collector and high performance database but also easy to use GUI for data visualization it can be used directly by data scientists without any customizations. Even though, in presented case, from user perspective, the performance of the system was excellent, next task would be to verify how this system work with multiple instances of Elasticsearch and how it will improve the performance. To improve the quality of visualization it is recommended to develop a few more dedicated plugins which will match exact requirements to visualize available data sources in described system.

REFERENCES

[1] International Telecommunication Union, ICT Facts and Figures 2016, http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2016.pdf (accessed 27.12.2016)
[2] Yi S.J., Chun S.D., Lee Y.D., Park S.J., Jung, S.H., Radio Protocols for LTE and LTE-Advanced, Wiley, 2012, page 11
[3] Buyya R., Dastjerdi A.V., Internet of Things: Principles and Paradigms, Elsevier, May 2016, page 3
[4] Lee I., The Internet of Things in the Modern Business Environment, IGI Global, 2017
[5] The Internet of Things: Mapping the Value Beyond the Hype, McKinsey Global Institute Analysis, June 2015
[6] Kranz M., Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry, Wiley, 2016
[7] Hype Cycle for Emerging Technologies 2015, Garner, August 2015
[8] Apache 2.0. license, http://www.apache.org/licenses/LICENSE-2.0 (accessed 01.01.2017)
[9] Procopiuc O., Agarwal P.K., Arge L., Vitter J.S. (2003) Bkd-Tree: A Dynamic Scalable kd-Tree. In: Hadzilacos T., Manolopoulos Y., Roddick J., Theodoridis Y. (eds) Advances in Spatial and Temporal Databases. SSTD 2003. Lecture Notes in Computer Science, vol 2750. Springer, Berlin, Heidelberg
[10] Elasticsearch 5.1 online documentation - Elasticsearch product description https://www.elastic.co/products/elasticsearch (accessed 01.01.2017)
[11] Elasticsearch in Anger: Stories from the GitHub Search Clusters https://www.elastic.co/elasticon/2015/sf/elasticsearch-in-anger-stories-from-the-github-search-clusters (accessed 01.01.2017)
[12] Stack Overflow Uses Facets and Geo-Coding https://www.elastic.co/videos/stack-overflow-uses-facets-and-geo-coding (accessed 01.01.2017)
[13] Kuć R., Rogozinski M., Elasticsearch Server, Packt Publishing Ltd, Feb 2016
[14] Paro A., ElasticSearch Cookbook - Second Edition, Packt Publishing Ltd, Jan 2015
[15] Elasticsearch 5.1 online documentation - Numeric datatypes, https://www.elastic.co/guide/en/elasticsearch/reference/current/number.html (accessed 01.01.2017)
[16] Es2Csv tool repository, https://github.com/taraslayshchuk/es2csv (accessed 01.01.2017)
[17] Ellingwood J., How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 16.04, https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-ubuntu-16-04 (accessed 01.01.2017)
[18] Search guard in a nutshell, https://floragunn.com/searchguard/ (accessed 10.01.2017)
[19] ReadonlyREST plugin, https://readonlyrest.com/ (accessed 10.01.2017)
[20] Roes T., Writing Kibana 4 Plugins, https://www.timroes.de/2015/12/02/writing-kibana-4-plugins-basics/ (accessed 10.01.2017)
[21] Kibana User Guide [5.2], Plugin development https://www.elastic.co/guide/en/kibana/current/plugin-development.html/ (accessed 10.01.2017)
[22] Broers R., Developing Kibana Plugins, https://amsterdam.luminis.eu/2016/04/07/developing-kibana-plugins/ (accessed 10.01.2017)
[23] Eclipse Paho JavaScript Client, https://eclipse.org/paho/clients/js/ (accessed 10.01.2017)
[24] Mosquitto, An Open Source MQTT v3.1/v3.1.1 Broker, https://mosquitto.org/
[25] Nof S.Y., Springer Handbook of Automation, Springer Science & Business Media, 2009
[26] Higginbotham S., Our connected future: What to expect when elevators and toys start phoning home, https://gigaom.com/2013/05/10/our-connected-future-what-to-expect-when-elevators-and-toys-start-phoning-home (accessed 01.01.2017)