MONITORING MIXED-LANGUAGE APPLICATIONS WITH ELASTICSEARCH, LOGSTASH AND KIBANA (ELK)

A. De Dios Fuente, O. O. Andreassen, C. Charrondière, CERN, Geneva, Switzerland

Abstract

Application logging and system diagnostics is nothing new. Ever since we had the first computers scientists and engineers have been storing information about their systems, making it easier to understand what is going on and, in case of failures, what went wrong. Unfortunately there are as many different standards as there are file formats, storage types, locations, operating systems, etc. Recent development in web technology and storage has made it much simpler to gather all the different information in one place and dynamically adapt the display. With the introduction of Logstash with Elasticsearch as a backend, we store, index and query data, making it possible to display and manipulate data in whatever form one wishes. With Kibana as a generic and modern web interface on top, the information can be adapted at will. In this paper we will show how we can process almost any type of structured or unstructured data source. We will also show how data can be visualised and customised on a per user basis and how the system scales when the data volume grows.

INTRODUCTION

At CERN, as in any other large organization, lots of data are generated and stored at a rate that makes it difficult for humans to analyse them. In addition, when considering the many different file formats, storage types, and physical locations, treating the data manually can become an unfeasible and overwhelming task. Manual treatment would go from data analysis to bug or error tracking. Having a tool that helps engineers understand the insights of the data stored, generate reports and gather statistics from them in a faster and easier way, is often the key to prevent future problems and where the challenge lies.

Moreover, with any kind of system, the logfile is the first place to look for clues when the system behaves in an unexpected manner. In our case, we have a variety of applications written in LabVIEW, C++ and Java, running on the most popular operating systems (Linux, Windows and OS X) and applications that make use of our LabVIEW Rapid Application Development Environment (RADE) [1], which has part of the services distributed over multiple servers and which is accessible from more than ~100 users at CERN. In such systems, most of the logs have different or weak structures. This led us to investigate how to unify all the different data formats and sources, and how to store all the information in a common place in order to have a tool that monitors and keeps track of all the online data easily and effectively in real time. However, since the needs of the users are different according to the application in hands, the way of getting statistics and reports should be customizable without web-development knowledge or specialised skills.

According to the motivation explained above, a conscientious study was made, in which several tools were identified, studied and tested.

STUDY OF TOOLS

One of the first things to consider was the data format. There is a large variety of sources; LabVIEW applications running on CompactRIO and PXI targets, Apache Tomcat servers, Java services, C++ applications and extensions, where each of them have different formats and purposes. In computing, syslog is a widely used standard for message logging [2] so this was the starting point to define the main format. In addition, the main requirements for the desired system are listed below:

- Support multiple log formats but mainly syslog.
- Support different communications and network protocols.
- Centralised data messages.
- Have a web-viewer to analyse the logs.
- Be able to get statistics of the stored data.
- Be fully compatible with Linux Environment.
- Be easy to use, install and configure.
- Be scalable if data grows over time.

Analysis of Logging Tools

Considering the advances in data mining and the fact that the analysis of "Big Data" [3] is not a new field, we decided to look for an already existing tool.

After searching and evaluating some promising tools found in the market, Logstash and Fluentd [4] were selected for a deeper study and test. Since these tools are evolving and have been redesigned several times due to their increasing popularity, some characteristics and features might have changed after our initial test. The versions used were Logstash 1.3, Fluentd 1.1, Elasticsearch 0.90 and Kibana 3.

Logstash and Fluentd are aimed to unify and manage data collection for better use and understanding. Both are free, open source and based in plugin models to extend functionality. The two of them can be combined with the popular Elasticsearch as the backend data store and Kibana as a front-end reporting tool to complete all the requirements.

Logstash is based on inputs, filters, codecs and output plugins where the inputs are the sources of the data, the filters are processing actions on the data under certain conditions, the codecs change the data representation and finally the outputs are the destinations where the data is sent. Fluentd has the same behaviour for the inputs and outputs but internally all the data are converted to JSON

ISBN 978-3-95450-148-9

(JavaScript Object Notation) [5] in order to give structure to an unstructured log message.

As a general comparison (Table 1), the two projects have similar features and capabilities; the most significant difference is that Fluentd insists on simplicity, versatility and robustness whereas Logstash focuses on flexibility and interoperability.

Table 1: Comparison between Logstash and Fluentd

Name	Logstash	Fluentd
Inputs/Outputs	Has ~30 inputs and ~50 outputs.	By default has ~10 inputs and outputs.
Filters	Parse logs into different formats	Not possible
Platforms	Any Java VM compatible platform	Not in Windows
Installation	All embedded in a unique jar	Need to install modules separately
JVM	Needed	Not needed
Data storage	Meant to work with Elasticsearch	Works with Elasticsearch, MongoDB…

Initial Testing

As a proof of concept both were installed and intensively tested to be sure about the amount of data they can receive and how they perform.

The conceptual tests were performed on two SLC 6 x64 based machines. One machine was used for Logstash and the other for Fluentd. The installation was really straightforward; it took less than a day with the basic configuration. Then, for Logstash, the following inputs were included: ZeroMQ [6], log4j [7] and UDP. ZeroMQ was used in the CERN middleware libraries for distributed messaging, log4j was the common Java logging utility and UPD was the internal transport protocol used by syslog. For Fluentd, we included the ZeroMQ and UDP inputs, but for Java logging we used its own implementation of log4j: fluent-logger-java.

Therefore, small test examples were implemented in C++ and Java using these protocols where the messages were sent from a different machine to the Logstash and fluentd instances. All machines were in the same network with a Gigabit Ethernet connection. For these tests, we sent 5000 messages per second to both instances and they worked seamlessly for several weeks in a row.

Finally, since both tools are really promising for our needs, we looked again into the rich collection of input and output plugins. Only Logstash supports RabbitMQ [8] (message broker software) and log4j, which are really useful for our National Instruments components and the Apache Server logs where our RADE Java libraries are running. Adding the fact of having all embedded into one JAR made it easier to install, so we selected Logstash as our collector and management of the log data.

ELK STACK

Although Logstash is a separate project, it has been built to work exceptionally well combined with Elasticsearch and Kibana. Together, known as the ELK stack, they become a powerful tool designed to search, analyse, and visualise data, allowing to get insight in real time [9].

ELK Architecture

The ELK stack is meant to be used in a distributed system where each component has different functionality and the usual architecture has five important components, which can be customised according to your system:

- Remote Shipper: collects logs from different machines and sends them to the central Logstash instance.
- Broker: a temporary buffer between the shippers and the central Logstash server. Typically, Redis is used as a queue-cache server, but also messages brokers can be used, such as RabbitMQ, ActiveMQ, etc.
- Indexer processes: they index and save the logs to the data storage.
- Elasticsearch: the storage and search engine.
- Kibana: the web interface.

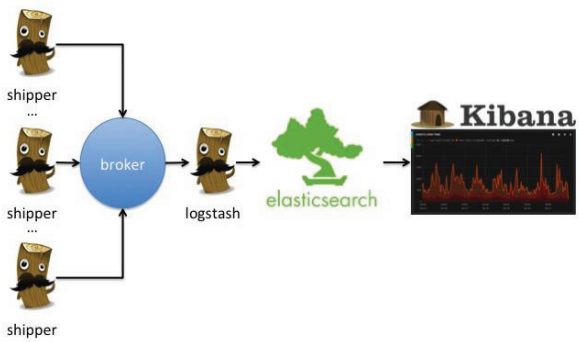

Figure 1: ELK Stack Architecture

Elasticsearch

Elasticsearch is based on Apache Lucene [10] that is the most powerful full-text search library available as open source and written completely in Java.

Moreover, it is a RESTful server so any action can be performed using its REST API using JSON over HTTP [11].

Elasticsearch provides an indexing service and a data storage without the need of defining a database scheme, but it is possible to provide one if needed.

It is meant for handling real time data that needs to be processed and analysed in a rapid manner. Also it is distributed and horizontally scalable which allows starting small and easily add nodes into the cluster if the data volume grows. The cluster consists of one or more nodes, which are established by adding a cluster name to each

ISBN 978-3-95450-148-9

node. One node in the cluster is elected to be the master node, which is in charge of managing the cluster changes. The data are stored in an index and are composed of a series of fields. Indexes are similar to a database and are mapped shards that hold a slice of all the data in the index. Shards can be either a primary shard or a replica shard. A replica shard is a copy of a primary shard used to provide redundant copies of the data to avoid loss of data in case of failure of any of the nodes. The number of primary shards in an index remains the same since its creation time, but the number of replica shards can be changed at any time [12].

Kibana

Kibana is the web interface embedded into Elasticsearch that helps to understand large volumes of data and rapidly detect patterns or irregularities in them. It provides a flexible and easy way to create dashboards thanks to the widget-based interfaces, which contains a huge variety of charts, graphs or maps. Most of the data is showed as time-series based.

The interface is simple, friendly and intuitive, there is no need to have a web development background to start using it. Since the data and the purposes are different per user, each of them can customize a dashboard according to his needs.

Using Logstash filters, one can create tags according to certain criteria and then it is possible to search and identify messages in Kibana.

In Figure 2, messages were indexed and tagged by Logstash depending on the log message content, then they were filtered with Kibana to visualise the statistics and metrics.

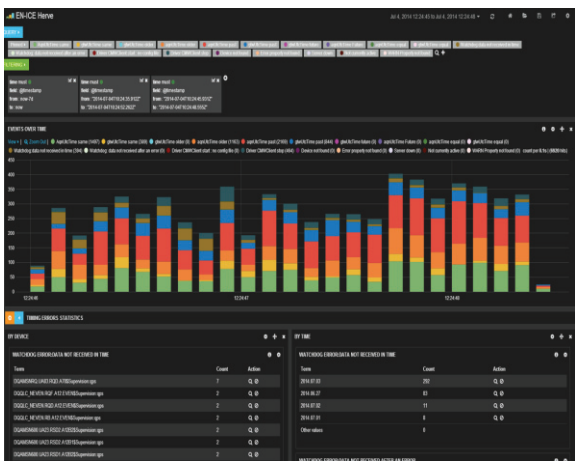

Figure 2: Kibana Dashboard

INTEGRATION, REMARKS AND ISSUES

After the initial tests, we analysed the data structure of the logs deeper and we decided to create in Logstash tags per host name, application name, error code, error log level and type of protocol used in order to identify messages and get statistics easily.

Moreover, we developed a Logger tool in LabVIEW where the messages are sent by UDP to the Logstash instance. The implementation of the utility took less than a day and it is included in the RADE framework where all LabVIEW/RADE users can benefit from it. As a result of this, it was possible to unify multiple log formats from different applications and users and, in some cases, give format to an unstructured data log.

In addition, the horizontal scalability feature that the ELK system provides was really useful. After months of use, we decided to add a new node, in order to avoid loss of data in case of failure. For doing this, it was needed to install a new instance of Elasticsearch in another computer and just give the same cluster name in the configuration file, Elasticsearch take care of the load balancing.

Also, at a certain moment, it was necessary to ship messages from CompactRIO at the Technical Network to the General Purpose Network where the ELK stack is installed. This was solved by adding a remote shipper, as was already explained earlier.

So for the bug diagnostics inside the RADE framework, especially for the Java services, it has been an important and valuable tool. Until now, when a problem appeared, we had to check hundreds of logs into several servers at multiple locations, spending a lot of time trying to find out where the error came from. Now, with the introduction of the ELK stack, a simple query can be run that will point to the problem instantly.

Although, the ELK stack offers many great features, it also has its downsides:

- The lack of security is one of its biggest defects. There is no access control to Kibana or ElasticSearch, so just knowing the URL of the host server, anyone with access to the machine can copy, modify or delete the dashboard or, even worse, the index directly from the database.
- Logstash has a limitation of its buffering capabilities, so if the number of messages keeps increasing, the internal buffer can fill up. In order to solve this, it is necessary to add a broker instance to hold and queue the events.

CONCLUSION AND FUTURE IMPROVEMENTS

Thanks to the introduction of the ELK stack, all the log messages have been unified into a common format and the data storage is centralised. The management and analysis of all these data has greatly improved, users have created their own dashboard according to their needs.

The bug diagnostics has been improved a lot thanks to the ELK stack; all the data logs are centralised in a single application and errors can be identified easily.

The time the developers spent identifying bugs under the RADE framework has been reduced.

One of the improvements we have in mind is to add an access control to avoid interactions of one user's

ISBN 978-3-95450-148-9

dashboards with other users' dashboards and also to add a security layer on top of all the stored data.

We are also planning to add new graphical components in Kibana and to extend it to be used in other websites outside the ELK stack.

REFERENCES

[1] O. Ø. Andreassen et al. "The LabVIEW RADE framework distributed architecture", ICALEPCS 2011, Grenoble, France, (2011)
[2] Definition Syslog: http://en.wikipedia.org
[3] Big Data: http://en.wikipedia.org/wiki/Big_data
[4] Fluentd website: http://www.fluentd.org/
[5] JSON website: http://json.org/
[6] ZeroMQ website: http://zeromq.org/
[7] log4j website: http://logging.apache.org/log4j/2.x/
[8] RabbitMQ website: https://www.rabbitmq.com/
[9] ELK products definition: https://www.elastic.co/products
[10] Lucene website: http://www.lucenetutorial.com/
[11] Elasticsearch website: https://www.elastic.co/products/elasticsearch
[12] C. Gormley, Z. Tong, "Elasticsearch: The Definitive Guide", (O'Reilly Media, Inc, 2015, 25-29)

ISBN 978-3-95450-148-9