Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

A Comparative Evaluation of Big Data Frameworks for Log Processing

Attila Péter Boros[a], Péter Lehotay-Kéry[ab], Attila Kiss[a*]

[a]ELTE Eötvös Loránd University, Budapest, Hungary
Faculty of Informatics, Department of Information Systems
attila9778@inf.elte.hu
lkp@caesar.elte.hu
kissae@ujs.sk

[b]Ericsson Hungary, Budapest, Hungary
peter.lehotay-kery@ericsson.com

Abstract

Nowadays a huge part of collected data comes from the behaviour of logging systems. Examples are complex monitored systems of different institutions where computations require powerful distributed environments to run. Our work aims the specific area of log data obtained from telecommunication operator systems with the goal to identify non-trivially detectable problems, like frequency of node restarts on a given time period or the reason of these events. In order to substitute significant new information from these system logs, it is important to use proper frameworks for analyzing them. This being a comprehensive problem, various frameworks have been proposed. In this paper we evaluate and compare Apache Spark and Elasticsearch (with Logstash) as two prominent frameworks for processing log data. Through our work we perform experiments on different problem solutions with different complexity in order to measure how non-functional features, like processing time and resource consumption vary between them. Additionally, our experimental data shows that how choosing between different frameworks can influence the performance of these computations.

Keywords: telecommunication, network, data analysis, Big Data, Elasticsearch, Spark, Log analysis

MSC: 68M14, 68W15

*A. Kiss was also with J. Selye University, Komárno, Slovakia.

1. Introduction

A system has been built, that aims on continuous, automatic software deployment on customer telecommunication network server nodes. Therefore, continuous automatic tests of new softwares and continuous automatic data collection have also been introduced for the network nodes.

At first the new software can be released on a small part of the live customer network, then the testing can be done on the field. If the new software works fine, the deployment can be extended on further nodes. It is also mandatory to see what are the configurations and states on the nodes before and after the upgrade and to see what events happened on the nodes after the upgrade. This part is done by the continuous data collection and processing on daily basis. Figure 1 shows the process.

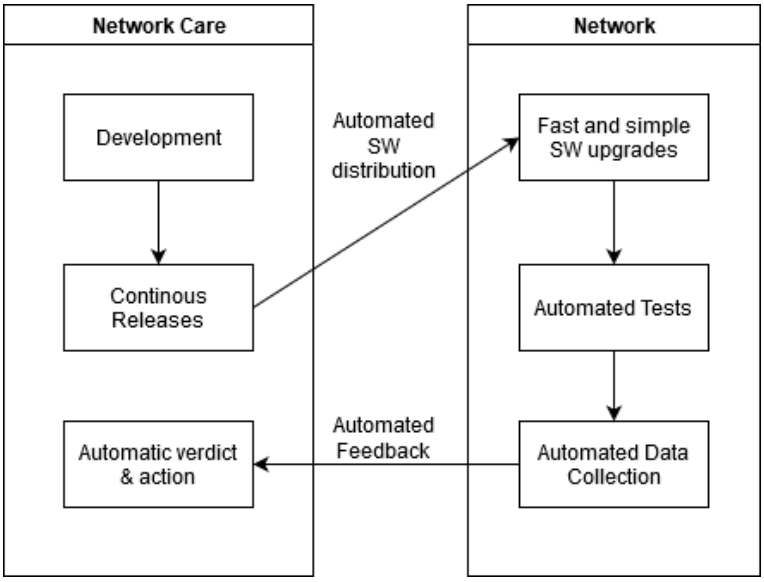

Figure 1: Development process

Thus we are aiming to collect node configuration-, state descriptor- and also log files generated by these nodes. This paper focuses on the processing of these log files, searching for restart events and also reasons behind these restarts, calculating logging intensity and the boot time, comparing the results before and after the upgrades. This gives us more insights on the impact of the upgrades and also we are able to react faster if needed, or in worst case scenarios to even call back the software upgrade.

Each customer network contains thousands of nodes and every node generates several kinds of logs. So for processing the large amount of files it is mandatory to use effective, distributed algorithms and technologies that support these kinds of

calculations.

First, we started with the processing of the configuration and state descriptor files. For these we used Spark, which fulfilled our needs. However, when it came to the processing of log files, we found that we should also investigate other technologies too. In this paper we are presenting our findings using Apache Spark and Elasticsearch.

2. Related works

In this section, first we look through what have been done in comparing technologies for log parsing.

In *Tools and benchmarks for automated log parsing*[1] authors evaluated 13 log parsers on a total of 16 log datasets spanning different architectures. They reported the benchmarking results in terms of accuracy, robustness and efficiency. They also shared the success stories and lessons learned in an industrial application.

Authors of *An evaluation study on log parsing and its use in log mining*[2] studied four log parsers and packaged them into a toolkit to allow their reuse. Also, by evaluating the performance of the log parsers on over ten million raw log messages in five datasets, they obtained insightful findings, while their effectiveness on a real-world log mining task has been thoroughly examined.

Now, let us look through what have been done using one of our chosen framework: Spark.

In *Log-based abnormal task detection and root cause analysis for spark*[3], authors proposed an approach to detect abnormality and analyze root causes using Spark on log files. Their proposed method has been tested on real-world Spark benchmarks.

Authors of *LADRA: Log-based abnormal task detection and root-cause analysis in big data processing with Spark*[4] proposed a tool, named LADRA, for log-based abnormal tasks detection and root-cause analysis using Spark logs. In LADRA, a log parser first converts raw log files into structured data and extracts features. Then, a detection method is proposed to detect where and when abnormal tasks happen. At last, leverage General Regression Neural Network (GRNN) to identify root causes for abnormal tasks.

Finally, let us look through what have been done using our other chosen framework: Elasticsearch.

Authors of *Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem*[5] used the Elasticsearch, Logstash and Kibana stack to set up a monitoring system to inspect the site activities. They fed heterogeneous accounting information to different MySQL databases and sent to Elasticsearch via a custom Logstash plugin. Then they were starting to consider dismissing the intermediate level provided by the SQL database and evaluating a NoSQL option as a uniquecentral database for all the monitoring information.

In *Elasticsearch and Carrot2-Based Log Analytics and Management*[6], authors reflected on how Elasticsearch along with Carrot2 is used with their algorithm to

manage and analyze logs of any format. They set up log analytics and management on Amazon web server.

3. Used technologies

Through our work Apache Spark and Elasticsearch combined with Logstash has been used.

Apache Spark[7] is a platform to implement large-scale data processing by providing a simple interface for efficient distributed computations on significant amount of data. It provides and includes useful basic functionalities like scheduling, distributing and data monitoring, while providing flexibility in integration with other frameworks. It provides an interface of several functional style methods with which we can operate on its resilient distributed datasets, which therefore can be distributed in an easier way. Spark also has support for efficient graph processing and machine learning libraries, providing them an efficient platform to distribute calculations i.e. when optimizing on different hyper-parameters while training neural networks.

Elasticsearch[8] is an open source search and analytic engine which provides a distributed framework handling all types of data. Being a standalone search engine, most of the time it serves as a core part of a layered monitoring system in real world applications, where to provide other services i.e. for presenting and persisting data, other plugins have to be attached. Logstash[9] serves as an extension plugin which strengthens the core API with the ability of preprocessing and delivering data read from different sources like local file system, databases and streaming applications etc. In the following we will refer to this set-up of Elasticsearch with Logstash plugin as ESL-stack. Being an open-source software it has developed a large and responsive community over time, which favors all developers from beginners to experts.

In the course of our test cases, we used Apache Spark 2.4.3, and Elasticsearch 7.5.2 with the same version of Logstash plugin. To provide equal scenarios for testing we seized the opportunity of ingesting data from local storage, in the same time eliminating the overhead of dealing with other frameworks related to the persistence of data. This direction of focusing only on the core system was also supported by the feature of Apache Spark, this framework being able to load inputs on its executors from local storage from version 2.0.0.

In such circumstances we were able to provide test cases which realize the same train of thought, expressed in the corresponding environment.

4. System description

For evaluating our implementations we used two environments. One single node cluster, with a machine equipped with an Intel Core i7-6700HQ processor, 8Gb of DDR4-1866MHz RAM and 1Tb 5400rpm of local disk storage; and another with a

set of 20 connected machines, each having node equipped with an Intel Core series processor running at 2.5GHz, 16Gb RAM and 250Gb disk storage, where nodes were strongly connected between them. In our former single-node Apache Spark set-up, the framework was deployed as a single-node cluster, and in our single-node ESL-stack set-up Elasticsearch and Logstash were deployed on the single available node. In the latter environments we had two different set-ups, one for Apache Spark (1 driver node, 10 executor node, each allocated with 10Gb of RAM), and one for Elasticsearch-cluster (1 coordinator node with 4Gb RAM, 5 master + data nodes with 8Gb RAM and 5 Logstash nodes each with 8Gb of RAM). Through our work we firstly investigated in evaluating scenarios in a single-node environment. Further works include investigating in evaluation and comparison on multi-node cluster.

5. Results

We have investigated evaluation on four test cases: restart count from error log files (a), actual restarts with module, trigger entry and trigger action (b), system boot-up times (c), and local log intensity (d), all grouped by node and date.

Test case (a) parses special log files gathered from nodes, which contain information about various system events. It collects the count of restart events from all the files, each line in the result showing that on the specific date and specific node how many restart events happened. This information is being acquired by parsing each log line and deciding by pattern matching that the actual event is a restart event, and if so when and on which node it happened.

Test case (b) parses the same specific files as test case (a), and aims for collecting only restart events, but instead of event count specific attributes about restart events are gathered. These specific attributes are module (which module initiates the restart), trigger entry (what is the reason of the restart event, in most cases a system error code) and trigger action (what action was triggered by the error).

Test case (c) collects information about how much time took a node to boot up. It is a much lighter test case than the two previous, because there is no need to parse all of the log lines, because in the ending part of every log file there is stated this information. But there is possibility in each framework only to parse whole files, so the specific line has to be filtered, and the information parsed out of it. Thus the result contains in each of its lines the boot time value in seconds of every node and every day.

Test case (d) gathers local log intensity. It is done similarly to test case (a), but it differs in the files which are parsed. An extra complexity is added in the case with the fact that these log files are enriched by multi-line scripts too. So an extra filtering is needed for log files to obtain only the scheme-based lines. From these lines is then gathered the count of events classified by node and date.

We have run each test case with both of the frameworks.

Our single-node test cases were run in each case on 13 files, each containing between 50 and 5000 log lines. Test results are shown in Figure 2. We conclude,

that throughout our test cases Elasticsearch performed better in each test case.

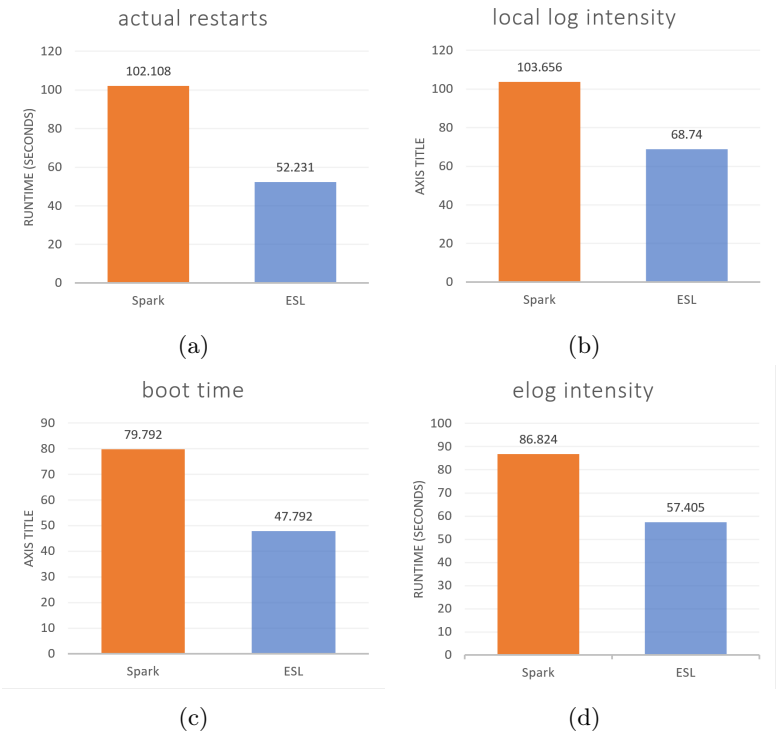

Figure 2: Comparison results of single-node evaluations

This could be explained by the fact that after pre-processing log file lines, the main search engine creates indices on documents received from Logstash, thus in further being able to perform faster searching. Meanwhile in Spark there is no searching, the output is a result of several transformation and aggregation on resilient distributed data-set (RDD) objects. Therefore another future work might include investigating in a search oriented, optimized Spark implementation, including the utilization of built-in data-frames and data-sets, from which the latter also includes a built-in query optimisation tool.

Further we set up and tested in multi-node environment our Spark implementations too. These test cases were run with the aforementioned setup, and on 1500 times more log files, which were generated from the basic sample. This means that in every test case 18000 log files were processed. The results of it are shown in Table 1.

Therefore as a future work we plan on comparing the two frameworks in real distributed environment with multiple nodes.

actual restarts	elog intensity	local log intensity	boot time
1571.4s	1534.26s	1260.463s	1440.710s

Table 1: Evaluation of Apache Spark test cases on multi-node environment

6. Conclusion

Through our work we targeted the comparative evaluation of Apache Spark and ESL-stack frameworks. As a result of our single-node cluster tests we came to the conclusion that in our cases the two computation environments perform differently on the test cases. After our evaluation, it came out that Elasticsearch performed significantly better, than Apache Spark. But the reality is, that each of these test scenarios could be improved.

Although this work serves as a proper foundation of future investigations in other similar comparative evaluations like testing our implementations in multi-node environments. Also our plans are elaborated more in detail in the Future Work section.

7. Future work

As a first task for our future works will be the testing of Elasticsearch in the defined test cases in multi-node environment. Therefore we will be able to compare the run-time performance of the two framework in real multi-node cluster environment.

Another future investigation will be on the side of security. The collected files can contain sensitive customer data, so we are planning to investigate more security solutions and encryption techniques, first checking the built-in solutions of the compared technologies.

It would also be good if we were able to predict the errors, so we are planning to develop some machine learning techniques to learn from the logs what kinds of events are happening before the errors and restarts.

Furthermore, we are planning to investigate the capabilities of Splunk and Flink frameworks too.

Acknowledgements. The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

This publication is the partial result of the Research & Development Operational Programme for the project "Modernisation and Improvement of Technical Infrastructure for Research and Development of J. Selye University in the Fields of Nanotechnology and Intelligent Space", ITMS 26210120042, co-funded by the European Regional Development Fund.

The project was also supported by the Ericsson-ELTE Software Technology Lab.

Furthermore thanks to Ericsson coworkers who worked on the project.

References

[1] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. Tools and benchmarks for automated log parsing, *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2019), 121–130.

[2] He, P., Zhu, J., He, S., Li, J., & Lyu, M. R. An evaluation study on log parsing and its use in log mining, *2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, (2016), 654–661.

[3] Lu, S., Rao, B., Wei, X., Tak, B., Wang, L., & Wang, L. Log-based abnormal task detection and root cause analysis for spark, *2017 IEEE International Conference on Web Services (ICWS)* (2017), 389–396.

[4] Lu, S., Wei, X., Rao, B., Tak, B., Wang, L., & Wang, L. LADRA: Log-based abnormal task detection and root-cause analysis in big data processing with Spark, *Future Generation Computer Systems*, Vol. 95 (2019), 392–403.

[5] Bagnasco, S., Berzano, D., Guarise, A., Lusso, S., Masera, M., & Vallero, S. Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem, *Journal of physics: Conference series*, Vol. 608 (2015), p. 012016.

[6] Singh, P. K., Suryawanshi, A., Gupta, S.,& Saindane, P. Elasticsearch and Carrot 2-Based Log Analytics and Management, *Innovations in computer science and engineering* (2016), 71–78.

[7] Frampton, Mike Mastering apache spark, *Packt Publishing Ltd*, Birmingham, UK (2015).

[8] Kuc, Rafal, and Marek Rogozinski Mastering elasticsearch, *Packt Publishing Ltd*, Birmingham, UK (2013).

[9] Turnbull, James The Logstash Book, *James Turnbull*, (2013).