Logging and Monitoring System for Streaming Data

Nguyen Ngoc Chung and Phan Duy Hung(✉)

FPT University, Hanoi, Vietnam
chung18mse13005@fsb.edu.vn, hungpd2@fe.edu.vn

Abstract. Logging and monitoring data is very important during the development process as well as the operation of information systems. As the data grows to TB every day, this problem becomes more complicated. Companies can generally buy big data analytics platforms or build it by themselves. Whether buying or building, it is important to have a realistic expectation of time and budget needed to successfully implement, roll out and provide ongoing support. There was a lot of confusion and frustration as the data platform market grew. Suppliers sell their capabilities instead of the actual needs of their customers. Contrary to that trend, some companies would like to build the platform using open source systems such as Apache Flume, Apache Spark Streaming and some other auxiliary technologies at a reasonable cost. This study analyzes requirements, introduces system architecture, and builds a logging and monitoring system for streaming data. The work is also a real project in the field of advertising.

Keywords: Log analysis · Monitoring · Apache flume · Apache spark streaming · Advertisement

1 Introduction

Collecting and monitoring the data generated during the development and operation of the systems can help the process of analysis, making strategic decisions for many companies. For example, during the operation of the system, statistics and descriptions of the failures allow early warning of infrastructure-related problems. Based on the warning content, the operator may have appropriate repair solutions so that the system can operate stably. For web systems, server metrics need to be monitored to allow automation scenarios for system scaling ability. The system will scale-out when high traffic and scale-in when low traffic. Another typical system that requires the continuous collection and monitoring of system parameters is the e-commerce system, managers need to see a user's log of actions taken on the website. Information that can be used to personalize user interfaces, predict and analyze user's behavior, etc. Moreover, when the data required to log increases up to TB every day, the analysis and visualization of data also arise a lot of complex problems.

This paper presents applied research for the Deha solution company's Adlink advertising system [1]. Adlink system includes modules such as advertising management, campaign management, user management and authorization, output tracking, and payment (Fig. 1). Tracking the effectiveness gained from each ad to optimize costs is very important. Managers need visual charts and recommendations, on which basis they can make timely decisions about whether to pause, resume ads, or even stop a campaign.

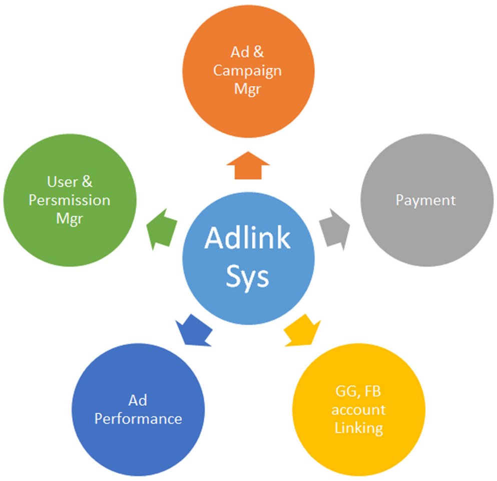

Fig. 1. Adlink system.

The system requires scalability as the number of customers changes can upload up to 10,000 ads to Google and Facebook. At the same time, the system needs to be able to monitor the effectiveness of these ads by querying periodically to Google, Facebook to get results. Do a simple calculation as follows: 10 k (ads) * 2 (request API per minute) * 4kbytes (size of API result) * 60 (min per h) ~ 4Gbytes data per hour.

Such a large amount of data requires a big data infrastructure that supports streaming data and does the two main tasks of collecting and providing monitoring tools for data analysis.

While researching solutions to build the solution for the Deha solution company, several related studies were found.

Some research related to service integration platforms by utilizing Apache Flume, Apache Spark Streaming. Apache Flume can be used to perform transfer logs, and Apache Spark Streaming can be used for capturing log and analyzing log. As in the study of real-time processing using Apache Flume [2], the authors study Flume's architecture and to be able to read and analyze Twitter data. The work shows that Apache Flume is well suited for real-time data processing. In another study, the authors also used Flume as a tool to collect and transfer huge data from Twitter to Hadoop's HDFS storage system, which is used to evaluate the number of tweets about candidates in 2017 Ecuadorian presidential election [3]. On the side of Apache Spark Streaming, in the research paper "Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities" [4], the authors point out that Spark Streaming has good scaling out and can handle high throughput and suitable for loading, extracting and converting data. In addition to the above studies, the study of "Dynamic Hashtag Interactions and Recommendations: An Implementation Using Apache Spark Streaming and GraphX" [5] also shows that Spark Streaming can also be used in a machine learning problem, which could potentially be applied to our actual systems in the future.

The above studies confirm that the Hadoop ecosystem with Flume, Spark Streaming provides powerful tools to help build in-house infrastructure systems that can collect data from multiple sources, storing, monitoring and analyzing to extract the necessary information for managers and it is good for Adlink management problem.

The remainder of the paper will be organized as follows: Sect. 2 describes the system requirements, system design and implementation presented in Sect. 3, and finally, conclusions and perspectives are made in Sect. 4.

2 System Architecture and Requirements

2.1 Architecture Overview

With the visions of building a large data collection and monitoring system, capable of scaling out well, operating independently of the current business system, the system architecture is proposed as Fig. 2 follows:

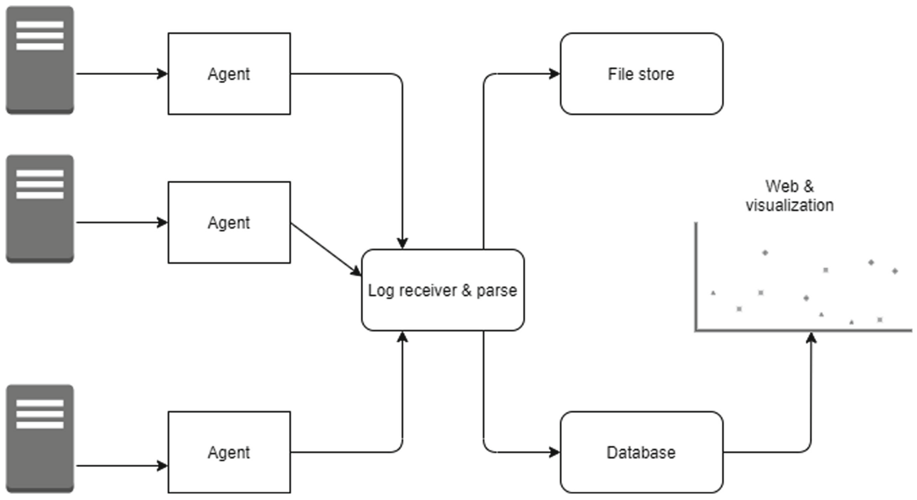

Fig. 2. System architecture

The system consists of main components as below:

- **Log Agents** are installed on application servers to collect and transfer log data to the Streaming Process component. These agents will use the configuration file generated by the Web Manager component.

- **Log receiver & parser component** performs real-time data analysis and saves aggregated data into the database.
- **Storage component** saves log data to file system for reuse when needed.
- **Database component** stores data after analysis which will be used for the Visualization component.
- **Web Manager & visualization component** presents data in the form of graphs, trigger warning base on predefined conditions.

2.2 System Requirements

The system has two main types of users that are system normal users, administrators. The functional requirements of each user are described in Table 1.

Table 1. Functional requirements.

Type	Content
Administrator	Manage User & privilege
	Generate and store configuration file for agents following predefined templates
	Login, Logout
Normal user	View dashboard
	View visual chart report
	Installation & Starter: Download and install the necessary libraries, predefined profiles and start the log agent

After using the system to create configuration files for the Flume agent, users will use it to proceed with their server configuration. Then, when the Flume agent is started, the log data in the specified directories will be transferred to the streaming processing component (receiver & parser). In this module, the log streams will be analyzed and converted into data types that the visualization will use to display the graphs.

3 System Design and Implementation

3.1 Servers

Based on the system architecture as well as the technologies selected, server configurations for system deployment (production) are listed in Table 2. Note that the system is flexible and scalable because it is deployed on Cloud AWS, so the table gives the minimum list for the system to work.

3.2 Apache Spark Streaming

Apache Spark Streaming is an extension of the core Apache Spark API that provides scalable, high-throughput and fault-tolerant stream processing of live data streams [6].

Table 2. List of servers used in the system.

No	Configuration			Type	Number	Function	Environment
	CPU (Cores)	RAM (Gb)	HDD (Gb)				
1	4	16	256	Virtual	1	Hadoop NameNode	Production
2	4	8	10240	Virtual	3	Hadoop DataNode	Production
3	4	16	1024	Virtual	1	Database	Production
4	4	8	100	Virtual	1	Web	Production

Data ingestion can be done from many sources like Kafka, Apache Flume, Amazon Kinesis or TCP sockets and processing can be done using complex algorithms that are expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to file systems, databases and live dashboards. Its key abstraction is Apache Spark Discretized Stream or, in short, a Spark DStream, which represents a stream of data divided into small batches. DStreams are built on Spark RDDs, Spark's core data abstraction. This allows Streaming in Spark to seamlessly integrate with any other Apache Spark components like Spark MLlib and Spark SQL.

3.3 Apache Flume

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data [7]. It has a simple and flexible architecture based on streaming data flows. It is robust and faults tolerant with tunable reliability mechanisms and many failovers and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

In this work, Apache Flume is used for transfer logs to HDFS as well as Apache Spark Streaming (Fig. 3). Just a simple configuration with a few steps is okay. This helps to build the system mainly in the analysis of log data.

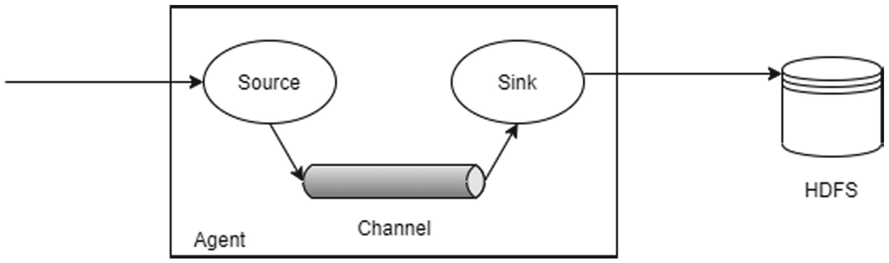

Fig. 3. Flume's operation.

The system uses a push approach, which means that the flume agent actively pushes the log onto Spark Streaming for processing log data (Fig. 4).

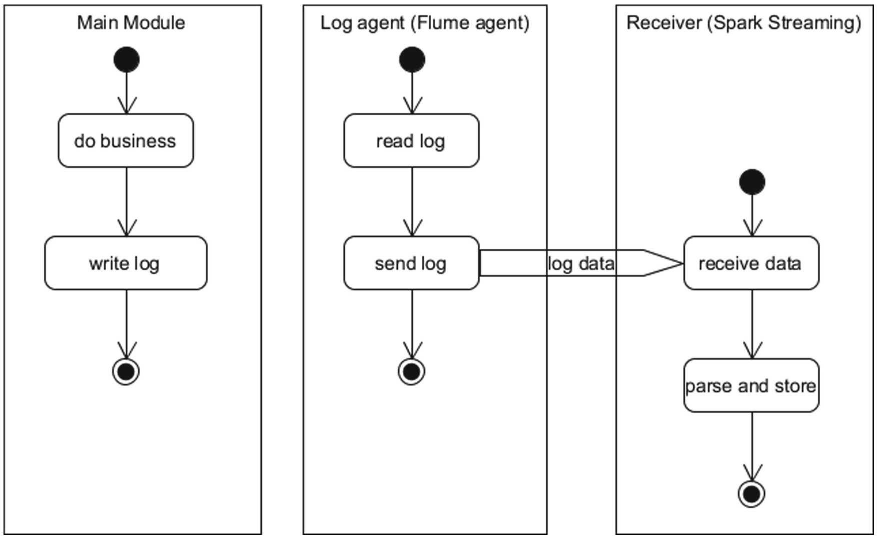

Fig. 4. Data processing.

According to this model, Flume Agent will actively read the log files in the specified folder and then transmit these log streams with Spark Streaming. The receiver then will analyze the log stream and continue to store data to the database or to HDFS.

3.4 Website Design

Regarding the representation of data in the form of graphs and charts, there are many tools like Apache Superset [8], Kibana [9], Grafana [10], etc.

All tools support various forms of data representation, but the data needs to be modified to suit the input format of each tool. This work has built Website management using PHP Laravel [11] and graphs using the Javascript library (ChartJS) [12].

Figures from 5 to 7 show the results after analyzing the log of different applications, including number of ad impressions, number of ad clicks, number of error transactions, number of successful transactions.

These results can help managers to make decisions. For example, Fig. 5 shows that the number of impressions for the ad with the orange line is significantly lower than the red line. If the cost for the orange line is higher than the cost for the red line, the corresponding ad to the orange line should stop. While from Fig. 6, on the 11th day, the output is low, Modifications to the image of the ad were subsequently made to increase the attractiveness. As a result, the number of clicks and the number of transactions increases.

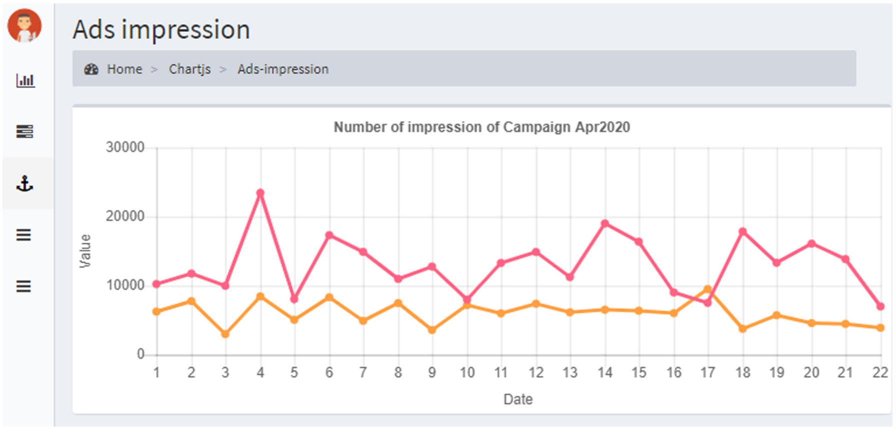

Fig. 5. Ads Impression.

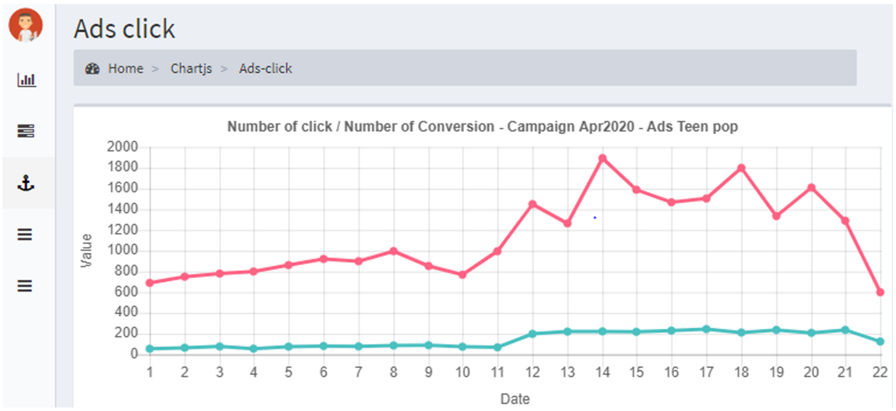

Fig. 6. Ads Click.

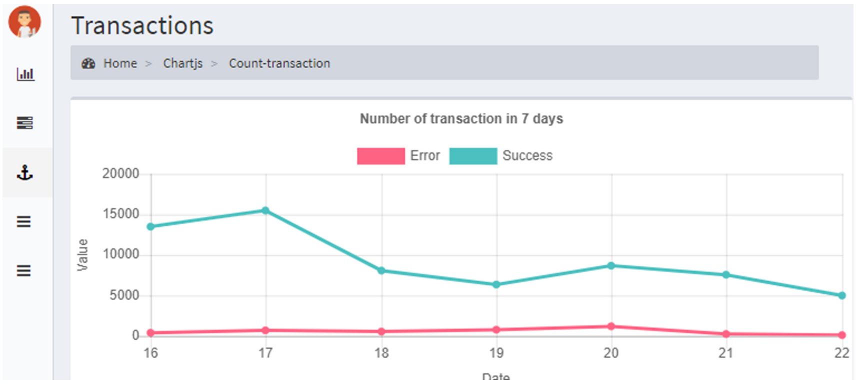

Fig. 7. Error Transaction vs Successful Transaction.

4 Conclusion and Perspectives

The paper presents the design and implementation of a system for logging and monitoring the streaming data. The system operates independently in parallel with the actual advertising system. The solution builds on open-source software Apache Flume, Apache Spark Streaming and some other open frameworks and libraries such as PHP Laravel, ChartJS, etc.

The system design allows for flexible scaling out. During the monitoring process, managers can rely on charts and analyzes to make strategic decisions. The system ensures the initiative of the business with a team of professional programmers available.

This solution is also fully applicable for similar company models and is also a good reference for research directions of Software engineering, Information System, etc.

References

1. Deha solution. https://deha.co.jp/
2. Srinivasa, K.G., Siddesh, G.M., Srinidhi, H.: Apache Flume, Network Data Analytics, pp 95–107 (2018). https://doi.org/10.1007/978-3-319-77800-6
3. Tenesaca-Luna, G.A., Imba D., Mora-Arciniegas, M.B., Segarra-Faggioni, V., Ramírez-Coronel, R.L.: Use of apache flume in the big data environment for processing and evaluation of the data quality of the twitter social network. In: Botto-Tobar, M., Barba-Maggi, L., González-Huerta, J., Villacrés-Cevallos, P., Gómez, O.S., Uvidia-Fassler, M. (eds) Information and Communication Technologies of Ecuador (TIC.EC). TICEC 2018. Advances in Intelligent Systems and Computing, vol. 884. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-02828-2_23
4. Nasiri, H., Nasehi, S., Goudarzi, M.: Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities. J. Big Data **6**(1), 1–24 (2019). https://doi.org/10.1186/s40537-019-0215-2
5. Sharma, S.: Dynamic hashtag interactions and recommendations: an implementation using apache spark streaming and GraphX. In: Sharma, N., Chakrabarti, A., Balas, V.E. (eds.) Data Management, Analytics and Innovation. AISC, vol. 1042, pp. 723–738. Springer, Singapore (2020). https://doi.org/10.1007/978-981-32-9949-8_51
6. Spark Streaming Programming Guide (2020). https://spark.apache.org/docs/latest/streaming-programming-guide.html
7. (2020). https://flume.apache.org/
8. (2020). https://superset.incubator.apache.org/
9. (2020). https://www.elastic.co/kibana
10. (2020). https://grafana.com/grafana/
11. (2020). https://laravel.com/docs/7.x
12. (2020). https://www.chartjs.org/docs/latest/