International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume 2, Issue 3 (March '2016)

# CLOUD BASED LOG STORAGE AND ARCHIVAL WITH REAL TIME SEARCH CAPABILITY

**Maria Michael Visuwasam,**
Head,
Department of Computer Science and Engineering
Velammal Institute of Technology,
Chennai,India.

**Abijith Krishna.U,**
Student,
Department of Computer Science and Engineering
Velammal Institute of Technology
Chennai,India.

**Balakumaran.M,**
Student,
Department of Computer science and Engineering
Velammal Institute of Technology
Chennai,India.

**Abstract:** Log data refers to a detailed list of application information, system performance, or user activities in a system with their respective timestamp. Logs are useful for keeping track of customer activities and understanding about the usage of the system. Logs can also be used for monitoring and ensuring the proper working of the system which can be reviewed chronologically. Systems usually store logs in MySQL databases and perform queries over them. They hugely rely on batch processing which does not support real-time search and analytics. Operations over the logs should be performed on real-time so that decisions and other improvements can be carried out as early as possible. Our proposed system ensures proper handling of logs, make them available for real-time search and store them in distributed environment. Proposed method uses Apache Kafka as message broker, Apache Hadoop HDFS for archival of log data and Elastic search to power near real-time search. Proposed system allows multiple clients to query log data, perform metrics and grouping analytics on them. We also make sure that logs are passed to the correct consumer through our message broker. Analytics over the log data helps us to visualize the pattern of usage of the system and its features with which we decide the performance improvements and business decisions. Further these can be represented as charts, graphs or tables for better visualization.

## I.INTRODUCTION

Generally Logs refers to a recording or footprint of activity done by a user while using a system. Usually Logs contain huge volume of data which are archived. Analyzing log data would provide a lot of information about the usage of the system and helps in tracking the user activities. It is a necessity to automate the process of aggregating log data, perform queries and other useful operations on them. In general log data includes fields like IP Address, HTTP Method, Time Zone, Timestamp, Status code, Client Details, Response Time, Operation Time and other custom fields.

One of the major problems in handling of logs is that huge amount of data will have to be pipelined to the correct consumer without any delay. This should be done with maximum accuracy as no log can be left unwanted. Any single log data could be crucial, which has a major effect on decision making or finding the error that occurred during the usage of the system. In addition to this, logs should be indexed to the near real-timesearch engine platform with very low latency. To satisfy all the above said conditions we require a scalable and distributed message broker service.

Consider a case of commodity price prediction. To predict the price of commodity we require historical price of the commodity over time and current price of the commodity. The more available and accurate the values of price of the commodity in both historical and present scenario, the more accurate would be the prediction made.

## II. EXISTING SYSTEM

Generally logs were stored even without any message brokers in MySQL databases. This caused a heavy overhead on the database server and this system is more prone to failures. Chances of data loss are very high in this method. Moreover queries were directly proportional to the size of the data stored. This was a huge problem as the size of log data may even grow exponentially and some queries will never return any useful information. The state of deadlock may occur due to use of very complex queries. When working with multiple tables and databases, manual joins were required which were tedious and could lead to human errors.

MySQL uses lock mechanism for concurrency which becomes a hindrance when multiple clients need to query the same data or when both indexing and querying occurs at the same time.The Logs can be from different sources. So having a specific schema for log data is practically impossible. Moreover schema altering queries are generally slow queries as they require reindexing of whole datasets. MySQL queries and other functions generally rely on batch processing which is not suitable for real-time analytics. While querying log data, users cannot be sure or accurate about the values they need. There should be possible matching terms within the possible editable distance to facilitate the log data search and analytics. MySQL does not

 

have native support for fuzzy queries. Figure 1 shows the architecture of the systems that are generally used for log data processing.

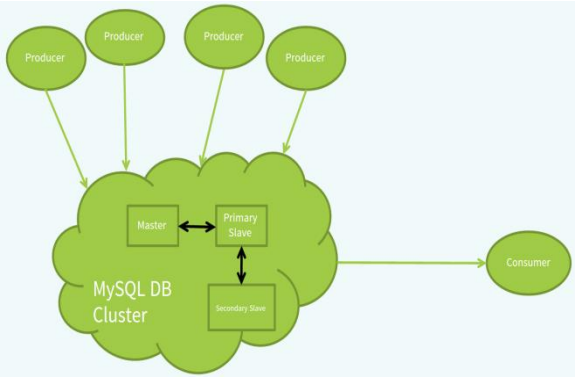


**Figure 1. Existing System Architecture**

## III. PROPOSED METHODOLOGY

The proposed methodology has two main modules which in turn contains multiple sub modules within them.

### Log Collection Module

For the analysis and aggregation process of the log data to occur, we must collect data from multiple clients concurrently. The logs are collected from various clients using agent systems or cloud storage facilities.

### Log Processing Module

This phase deals with the processing of logs. This can be done with both batch processing or stream processing. Both strategies have their own advantages and disadvantages for developing a real-time platform.

### Log Storage Module

Though logs will be indexed into our search engine, it is important that we store our log data for long term archival as there are chances of failure in search servers. As any log data could be valuable we perform both indexing log data to search server and storing it for archival at the same time concurrently. There are various methodologies for storing and processing real-time data in any system. They are as follows:

- Method 1: Store data to SQL databases
- Method 2: Store data to NoSQL databases
- Method 3: Directly index data to search engine server

### Log Visualization Module

Visualization of the results obtained by slicing and dicing of the logs from various systems can be done by using various visualization environments. In general we can use web applications for visualization as it would be less platform dependent.

### Audit Client

The only operation performed at client side is collecting logs and sending it to the server side. As we require real-time processing, logs that are pulled from the target application during the usage of the system itself. Java Message Service (JMS) API is used to get the log data from the client. The main advantages of JMS API is that it is loosely coupled and uses asynchronous communication. There are two major types of JMS message domains. We generally use point to point, as overhead on the client is reduced. Client activities are captured and sent to Queue. Queues are responsible for holding and delivering the log data from the clients using the application to the system present at server side. The client at the server side keeps listening to various queues from which logs are collected. The recognized fields along with the other unknown fields are joined into a single JSON (Javascript Object Notation) object.

### Audit Server

Kafka is run on an Apache Zookeeper cluster. Zookeeper performs various operations in the cluster such as data distribution, offset storage and many other cluster based operations. Kafka maintains feeds of messages in categories called topics. For ease of use and to prevent more failures Kafka can be deployed in a separate distributed environment. Configurations and topic creation are done once. In this way managing the Kafka cluster is easier as the Kafka cluster is loosely coupled to the whole system. The Kafka producers get log data from multiple JMS queues.

These are then put into kafka topics in the message broker cluster. These are consumed by single or multiple consumers can parallelly read messages from various topic partitions. The kafka consumer writes the log data into Apache Hadoop HDFS (Hadoop Distributed File System) as well as index the log data into Elasticsearch server. The data is written as map files into HDFS to ensure high availability, which can retrieved if necessary. HDFS can hold large volumes of data, so logs for very long period of time can be stored without much over head. It follows the master-slave architecture.

The logs indexed to the elastic search index can be used to perform various queries and aggregations. Elasticsearch is a search server based on Lucene. Indices can be divided into shards and each shard can have zero or more replicas. The kafka producers get logs from JMS client which is decoded into a normal JSON object. This JSON object is put into Hadoop as MapFile. Simultaneously the log data in the form of JSON object is indexed to the Elasticsearch search server. Thus the log data is available for near real-time search on which any query or aggregations can be done with help of Query DSL. Figure 2 shows the architectural overview of the proposed methodology.

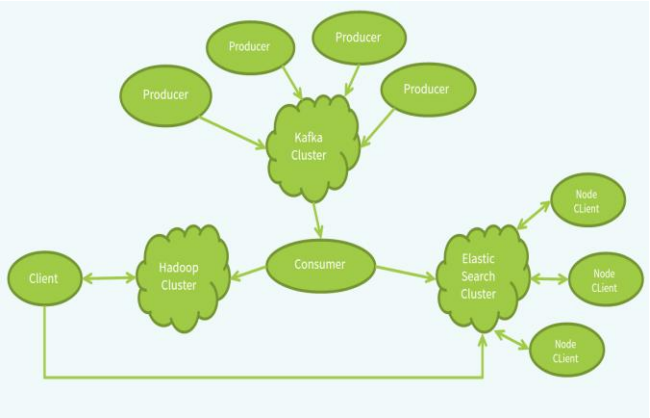


**Figure 2: Architectural Overview of the proposed system.**

Consider a situation in which the search server cluster is rebooted or needs some upgradation. In these cases the logs are only stored to the HDFS. After the search server cluster is back to normal, an HDFS client can decode from a MapFile to a JSON object and index them back to the Elasticsearch cluster. Logs are periodically removed from the Elasticsearch index. During particular scenarios where even older logs are required for computation, the HDFS client can read those logs from the Hadoop cluster, reindex them back to the search server and make them available for near real-timesearch.

## IV. ANALYSIS

The processing deployment was done locally while making comparison for the primary used data store i.e. MySQL and proposed log visualization layer i.e. Elasticsearch. We generally considered the time taken for indexing the logs.Figure 3 shows the performance comparison between the both. We can clearly see that MySQL becomes unstable and has many sporadic spikes for larger data-sets while in Elasticsearch the cluster tends to be stable and even ready to be queried. The proposed platform also had many added advantages as logs can be queried based on the time they are created. The can be routed to the specific indices based on their time-stamp value. This also makes sure that while querying we have low latency and minimal seek time for the data to be queried and aggregated.

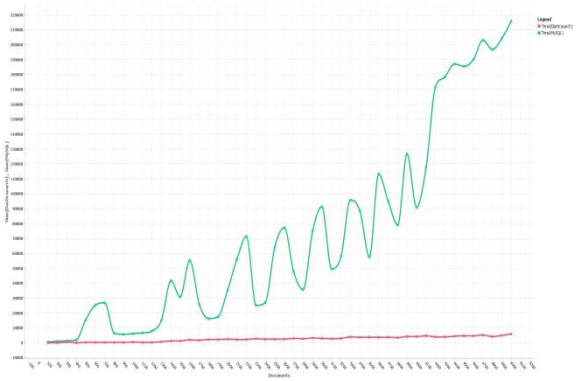

**Figure 3: Comparison of indexing time between MySQL and Elasticsearch for various number of logs.**

## V. CONCLUSION

The usual methodology of logs storage and retrieval is store the logs, process the stored logs in batch, again store the processed logs and run visualization queries and perform aggregations over these processed logs. But our methodology involves processing the logs during the message passing itself, index these processed logs to search engine indices and directly visualize them. This reduces the over head and overall latency. The platform can take various formats of logs based on the regular expression provided. The proposed methodology can also be extended as a base module for other application performing decision making and other functionalities based on the log data. The platform can be slightly altered to use any type of data other than logs for performing visualization operations.

## VI. REFRENCES

[1] Jandaeng C. Comparison of RDBMS and document oriented database in audit log analysis.IEEE 7th International Conference on Information Technology and Electrical Engineering (ICITEE) , pages 332-336, 2015.

[2] Srikrishnan V , Sivasankar E , Pitchiah R. A log data analytics based scheduling in open source cloud software. IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC), pages 390-395, 2014.

[3] Qi M , Liu Y , Lu L , Liu J , Li M. Big Data Management in Digital Forensics. IEEE 17th International Conference on Computational Science and Engineering (CSE), pages 238-243, 2014.

[4] Singh A, Vélez H.G. Hierarchical Multi-log Cloud-Based Search Engine. IEEE 8th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pages 211-219, 2014.

[5] Jun Bai. Feasibility analysis of big log data real time search based on Hbase and ElasticSearch. IEEE 9th International Conference on Natural Computation (ICNC), pages 1166-1170, 2013.

[6] Hingave H, Ingle R. An approach for MapReduce based log analysis using Hadoop. IEEE 2nd International Conference on Electronics and Communication Systems (ICECS), pages 1264-1268, 2015.

[7] Xiuqin Lin, Peng Wang, Bin Wu. Log analysis in cloud computing environment with Hadoop and Spark. IEEE 5th International Conference on Broadband Network & Multimedia Technology (IC-BNMT), pages 273-276, 2013.

[8] Jayathilake D. Towards structured log analysis. IEEE International Joint Conference on Computer Science and software engineering, pages 259–264, 2012.