Log Analytics using big data technologies

Executive Summary:

This report presents the findings of log analytics performed on the provided dataset. Section A focuses on log analytics using Apache Spark, while Section B incorporates a combination of Big Data technologies, including Kafka, Spark Streaming, Parquet files, and HDFS. Section C is an optional bonus task that involves developing a clustering algorithm for grouping requests based on various factors. The report highlights key insights, presents analysis results, and recommends further improvements.

Introduction:

This section provides an overview of the log analytics project and its objectives. It explains the significance of log analytics for the organization and outlines the scope and timeframe of the analysis.

Methodology:

The methodology section describes the approach and tools used for log analytics. It highlights the implementation of log analytics from the provided article and includes the code for '3.1' and '3.2' tasks. The data preprocessing steps, EDA, and log analysis techniques are also explained.

Log Analysis Findings:

4.1. Section A: Solving and Familiarizing with Log Analytics

The analysis focuses on generating output presenting the endpoint that received the highest number of invocations on a specific day of the week, along with the corresponding count of invocations for the entire dataset.
The analysis also calculates the total number of 404 status codes generated on each day of the week for the entire dataset.

4.2. Section B: Log Analytics Using Big Data Technologies

The analysis follows a flow starting with a Kafka producer, followed by a Kafka consumer performing transformations using Spark.

The transformed data is then converted into Parquet file format and stored in HDFS.
Key steps and techniques from the article are implemented to perform exploratory data analysis and obtain valuable insights.
Recommendations:
Based on the log analytics findings, the following recommendations are provided:

- Enhance the log collection process to ensure comprehensive coverage.
- Optimize performance by fine-tuning Spark Streaming and Kafka configurations.
- Implement real-time monitoring and alerting mechanisms for critical events.
- Apply further data preprocessing techniques to handle outliers and missing values effectively.
- Explore advanced machine learning algorithms for more advanced log analysis, such as anomaly detection or predictive maintenance.

Problem encountered:
We encountered several problems; below is a list of problems and how did we solve the problem.

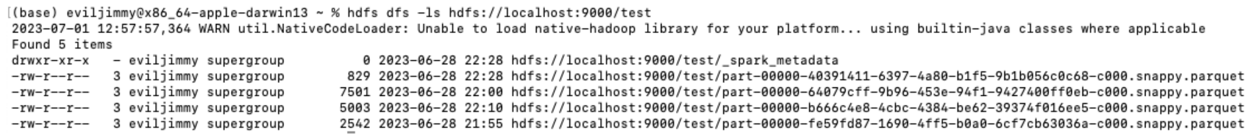
- Write the producer and consumer script using Python Kafka (confluence Kafka) and spark streaming API. By reading the document and testing the code, we have a better understanding of how each component of the streaming pipeline works together.
- Save the streaming data to HDFS. By understanding the structure of the data node and name node and debugging the HDFS error while launching in the terminal, we learned that we need to write the HDFS path correctly.
- Regex pattern to extract the desired fields from the log line. We tried different regex patterns to extract different patterns of the log files.

Conclusion:
The log analytics project using Apache Spark and Big Data technologies demonstrated the potential for extracting valuable insights from log data. By leveraging these technologies, the organization can enhance its decision-making processes, improve system performance, and strengthen security measures. Implementing the bonus task showcased the ability to cluster streaming messages based on selected criteria. Further enhancements and refinements can be made to optimize the log analytics process and maximize its benefits.

Appendices:

Source code: Include the implemented code from Section A and Section B.https://github.com/ZCai25/de_hw_kafka_spark/tree/main/kafka_test

Output screenshots: Provide relevant screenshots of output and analysis results.



```
(base) eviljimmy@x86_64-apple-darwin13 ~ % hdfs dfs -ls hdfs://localhost:9000/test
2023-07-01 12:57:57,364 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 5 items
drwxr-xr-x   - eviljimmy supergroup          0 2023-06-28 22:28 hdfs://localhost:9000/test/_spark_metadata
-rw-r--r--   3 eviljimmy supergroup        829 2023-06-28 22:28 hdfs://localhost:9000/test/part-00000-40391411-6397-4a80-b1f5-9b1b056c0c68-c000.snappy.parquet
-rw-r--r--   3 eviljimmy supergroup       7501 2023-06-28 22:00 hdfs://localhost:9000/test/part-00000-64079cff-9b96-453e-94f1-9427400ff0eb-c000.snappy.parquet
-rw-r--r--   3 eviljimmy supergroup       5003 2023-06-28 22:10 hdfs://localhost:9000/test/part-00000-b666c4e8-4cbc-4384-be62-39374f016ee5-c000.snappy.parquet
-rw-r--r--   3 eviljimmy supergroup       2542 2023-06-28 21:55 hdfs://localhost:9000/test/part-00000-fe59fd87-1690-4ff5-b0a0-6cf7cb63036a-c000.snappy.parquet
```