Detailed Project Report: Sentiment Analysis Pipeline Using HDFS and Spark

CONTENTS:

* Abstract………………………………………………………………..3
* Objective………………………………………………………………3
* Introduction………………………………………………………….4
* Problem Statement……………………………………………….4
* Architecture…………………………………………………………..5
* Design Details………………………………………………………..5
* Detailed steps………………………………………………………..6
* Benefits………………………………………………………………….10
* Conclusion……………………………………………………………..10

**Abstract**

Sentiment analysis and textual emotion recognition are closely related. Sentiment analysis is target-oriented, aiming to identify opinions or attitudes towards topics or entities (e.g., product, movie, Customer Reviews etc…). Emotion recognition, on the other hand, focuses on recognizing either the emotion expressed in text or evoked by the text, with no attachment to a specific target.

Sentiment analysis or opinion mining is the computational study of people’s opinions, sentiments, attitudes and emotions expressed in written language. It is one of the most active research areas in natural language processing and text mining in recent years.

**Objective**

The main objective of this project is to design a scalable pipeline using Spark to read customer reviews from the S3 bucket and store them into HDFS. The pipeline will be scheduled to run iteratively after each hour, and Spark Machine Learning will be used to perform sentiment analysis on the customer reviews and need to store in HDFS for further analysis using Hive/Hue.

**Introduction**

Our goal is to develop a scalable and efficient pipeline for performing sentiment analysis on customer reviews using Spark. The pipeline will be designed to read data from an S3 bucket, and store it in HDFS, and then need to perform sentiment analysis on the data using Spark Machine Learning. The output of the above analysis will be stored back in HDFS for further analysis.

**Problem statement**

Design scalable pipeline using spark to read customer review from s3 bucket and store it into HDFS. Schedule your pipeline to run iteratively after each hour.

Create a folder in the s3 bucket where customer reviews in json format can be uploaded. The Scheduled big data pipeline will be triggered manually or automatically to read data from The S3 bucket and dump it into HDFS.

Use Spark Machine learning to perform sentiment analysis using customer review stores in HDFS.

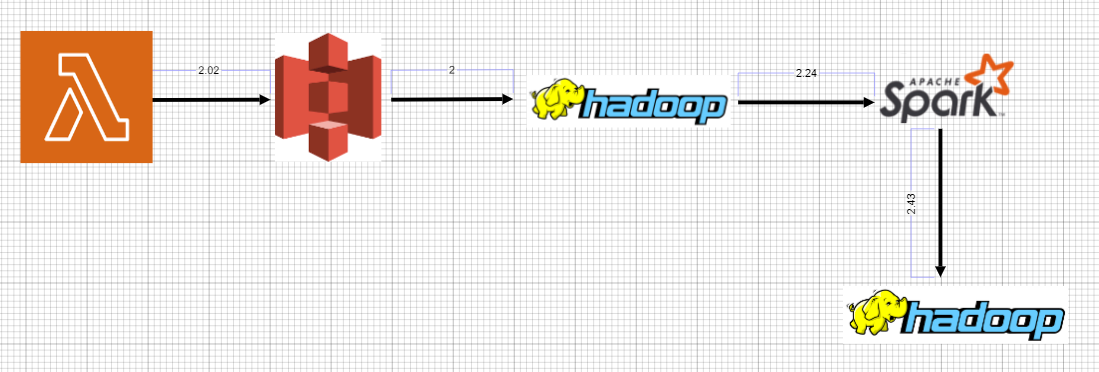
**Architecture**

The architecture of the proposed pipeline consists of three main components:

Data Source: The pipeline will read customer reviews data from an S3 bucket. This data will be in JSON format.

Processing Engine: Spark will be used to process the data, perform sentiment analysis using ML libraries, and store the output in HDFS.

Storage: The final output of the pipeline, which includes sentiment analysis results, will be stored in HDFS.



**Design Details**

Data Source: The pipeline will read customer reviews data from an S3 bucket. This data will be in JSON format.

We will upload a file to s3 which is in csv format. By using Lambda functions we are changing the format from csv to json and saving the json file in different folder in same s3 bucket.

For every 1 hour lambda function will get triggered by event bridge.

**Processing Engine:**

Once the data is uploaded to the S3 bucket, the pipeline will be triggered manually or automatically to read data from the S3 bucket and dump it into HDFS. We will use Spark to perform this operation.

The data will be processed using Spark, which is a fast and distributed processing engine for big data. We will use Spark’s Machine Learning Library (ML libraries) for performing sentiment analysis on the customer reviews.

To execute the pipeline, we will create a Spark job in Python. The job will be responsible for reading data from the S3 bucket, performing sentiment analysis using ML libraries, and storing the output in HDFS.

**Storage:**

The final output of the pipeline, which includes sentiment analysis results, will be stored in HDFS. The output can be analysed further using other big data tools such as Apache Hive, Hue…

**Detailed Steps**

**AWS S3:**

1.Create a folder in the S3 bucket.

2.Upload the csv file.(sentiment-anal-intern26/data.csv)

**AWS Lambda:**

1.Search the lambda in the AWS console

2.Click the "Create function" button.

3.Select "Author from scratch" and enter the name of the function.

4.Choose "Python 3.8" as the runtime.

5.Under "Permissions", create a new role with S3 policy full access.

6.Click "Create function".

Once created. Write the code. Click on deploy and test.

Check logs in cloud watch.

**Event Bridge**:

Create a scheduled job to trigger the pipeline

To trigger the pipeline at regular intervals, we will create a scheduled job using AWS Event Bridge. The job will run every hour and trigger the Lambda function to read the csv file from s3 bucket and convert into json and store in json location.

To create a scheduled job, follow these steps:

1.Go to Event bridge.

2.Select the Event bridge schedule.

3.Click the "Create rule" button.

4.Under "Event source", select "Schedule".

5.Enter a name for your rule and choose the "Fixed rate of" option.

6.Set the rate to 1 hour and click "Add target".

7.Choose the "Lambda function" option and select your Lambda function from the drop-down list.

8.Click "Configure details".

9.Review the settings and click "Create rule".

Once created, Every 1 hour it will run automatically. If any new data is received it will update.

**AWS CLI:**

Install the aws cli in local.

Once the AWS CLI is installed, configure it with the appropriate access keys to access the S3 bucket. You can use the command aws configure to configure the CLI with your access key ID and secret access key.

IAM-User

Enter name and mandatory details and download the file for credentials.

Procedure;

Run the docker compose file which has hadoop and pyspark images.

In your Dockerfile, install the AWS CLI and any other dependencies needed to interact with S3. For example, if you are using Python, you can add the following lines to your Dockerfile:  
RUN apt-get update && apt-get install -y python3-pip  
RUN pip3 install awscli

Once the container is up and running, create a directory in HDFS using the following command:

Once the Docker image is built with the necessary dependencies, you can run a container using that image and use the AWS CLI to download the file from S3. Here's an example command to download a file named example.txt from an S3 bucket named my-bucket:  
  
docker run -v /path/to/host/folder:/path/to/container/folder my-image aws s3 cp s3://my-bucket/example.txt /path/to/container/folder

This command maps a volume from the host machine to the container so that the downloaded file can be accessed outside of the container.

Finally, once the file is downloaded, you can use Hadoop commands to move the file from the container to HDFS. For example, if you are using the Hadoop command line interface, you can use the hdfs dfs -put command to copy the file from the container to HDFS:

aws s3 cp s3://customer\_reviews/customer\_reviews.json /home/hadoop/

docker exec my-container hdfs dfs -put / home/hadoop/customer\_reviews.json /input/

Spark:

In this step, we will use Spark Machine Learning to perform sentiment analysis on the customer reviews. We will be using PySpark to run the Spark jobs in Python.

In this step, we will store the results of the sentiment analysis in HDFS. We will create a new directory in HDFS to store the results.

docker exec my-container hdfs dfs -mkdir /output

Use the following command to store the results of the sentiment analysis in HDFS:

result.write.mode("overwrite").format("csv").option("header", "true").save("hdfs://localhost:9000/output/results.csv")

The results will be available in hdfs.

**Benefits**

**Accuracy**: The use of Spark’s Machine Learning Library ensures accurate results.

**Real-time analysis**: The pipeline can be scheduled to run iteratively after each hour, which ensures real-time feedback based on the customer satisfaction.

**Scalability**: The proposed pipeline is highly scalable and can handle large volumes of data.

**Efficiency**: The use of Spark enables the pipeline to process and analyse the data efficiently, which leads to faster results.

**Reusability**: The code written in the components used should have the ability to be reused with no problems

**Application Compatibility**: The different components of the project will be using python as interface between them. Each component will have its own task to perform, and is the job of the python to ensure proper transfer of information.

**Resource Utilization:** when any task is performed, it will likely used all the processing power available until that function is finished.

**Conclusion**

Finally conclusion, The pipeline uses Spark to process the data, perform sentiment analysis using ML libraries, and store the output in HDFS. This pipeline provides real-time feedback on customer satisfaction and enables businesses to identify for improvements and analysis on the future. It also help to identify the imbalance and can take the necessary action on the errors/corrections to stop or rectify based on the customers. Spark-based Sentiment Analysis Pipeline for Customer Reviews is a scalable and efficient solution for analysing customer reviews data.