Low Level Design

Sentiment Analysis

Written by

Rajdeep Sonawane

Contents

1. Introduction

1.1 What is Low-Level design document?

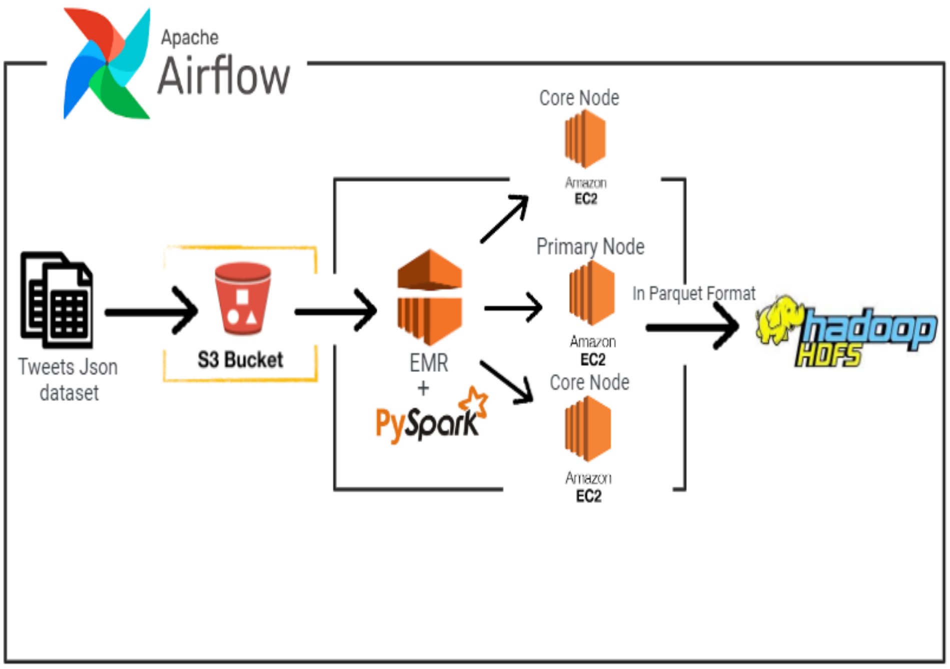
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

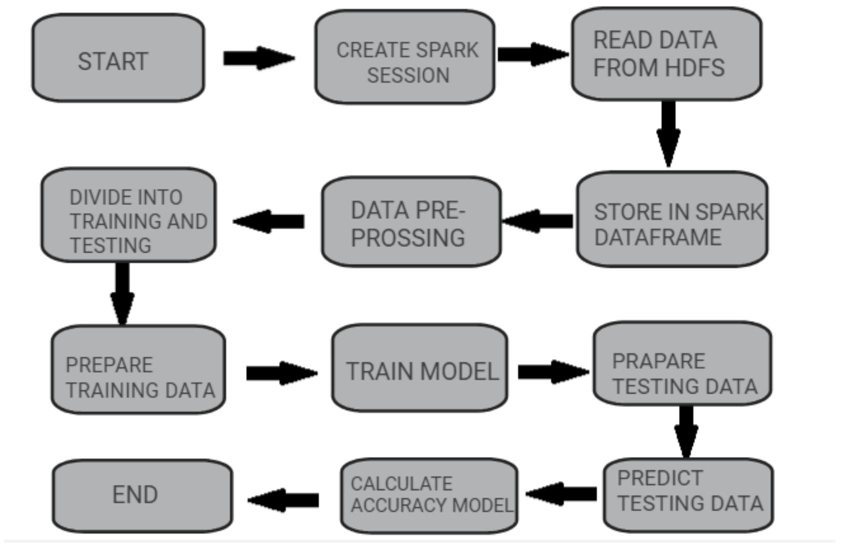
Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture

2.1 Scalable pipeline using spark to read customer review from s3 bucket and store it into HDFS



2.2 Sentiment analysis using spark ML



3. Architecture Description

3.1 Pipeline Components:

3.1.1 Data Source:

- **S3 Bucket**: Set up an Amazon S3 bucket to store customer reviews in JSON format. Create a dedicated folder within the bucket to organize the data. New reviews can be uploaded to this folder periodically.

3.1.2 Spark Cluster:

- **Master Node**: The master node manages the distribution of tasks across worker nodes. It coordinates the execution of the Spark application.
- **Worker Nodes**: Worker nodes are responsible for executing tasks assigned by the master node. They process the data in parallel, enabling efficient distributed computing.

3.1.3 Scheduler:

- **Apache Airflow** : Use a workflow management platform like Apache Airflow to schedule and orchestrate the pipeline. Define tasks and dependencies to ensure the pipeline runs smoothly and efficiently. Schedule the pipeline to run iteratively after each hour to process new customer reviews.

- **Cron Jobs**: Alternatively, you can use cron jobs on the server hosting the Spark cluster to trigger the pipeline at specified intervals.

3.1.4 Data Storage:

- **HDFS**: Hadoop Distributed File System (HDFS) is used for storing the processed data. Write the original customer reviews, to HDFS.
- **Parquet Format**: Store the data in Parquet format, which is efficient for columnar storage and supports compression. Partition the data based on relevant criteria such as date or product category for optimized querying.

3.1.5 Data Processing:

- **Spark Application**: Develop a PySpark application using Python to read data from the S3 bucket and to store in HDFS.

Preprocessing: Preprocess the raw text data to clean and tokenize it. This may involve removing punctuation, converting text to lowercase, and tokenizing sentences into words.

- **Stop Words Removal**: Remove common stop words (e.g., "the", "is", "and") from the text to reduce noise in the sentiment analysis process.

3.1.6 Sentiment Analysis:

- **Spark MLlib / Spark NLP**: Utilize Spark's machine learning libraries such as MLlib or Spark NLP for sentiment analysis. Train a sentiment analysis model on labelled data or use pre-trained models available in Spark NLP.

3.2 Pipeline Workflow:

3.2.1 Data ingestion to S3:

Create a folder in S3 bucket where Json data can be stored .The Json data contained the Customer Revive datasets. These datasets need to be transformed in parquet file for fast Processing .

3.2.2 Create a cluster:

Create a Cluster using Amazon EMR (ElasticMapReduce) . Using EMR we can create a Hadoop cluster . In Hadoop cluster we need primary node and secondary node for that we use Amazon EC2 instance. We use a EC2 instance for creating the primary node and secondary node. Attach the EMR cluster with EC2 instance . We have to create VPC for EC2 There we need to set the security groups , subnets and route tables. Use pyspark to read the Customer Revive Json data and store in HDFS in parquet file format. In addsteps we need to add the pyspark file to run the program or we can create the Scheduler using airflow which will schedule your pipeline to run iteratively after each hour.

3.2.3 Create a Scheduler:

Create a scheduler using Apache airflow to run iteratively after each hour . For that we can use Amazon Managed airflow or we can use container image from Docker .We need to install the Docker to run the docker image . For not getting charged from

Amazon Managed airflow we can install Vscode and by using compose.yml file we can create an airflow image inside the Vscode. For Scheduling the pipeline, we have to create a python file .In python file, we have to create a DAG(Directed Acyclic Graph). And in Dag we need to Specify the EMR configuration and the scheduling time we have to set the aws_default config. Using airflow UI, we can trigger the Dag.

3.2.4 Data Storage in HDFS:

The Json Data which is stored in S3 bucket after triggering the airflow Dag ,the data transformed in parquet file and store in HDFS for fast processing . Using these parquet data we need to perform Sentiment Analysis.

3.2.5 Sentiment Analysis using Spark ML:

Spark MLlib : Utilize Spark's machine learning libraries such as MLlib for sentiment analysis. Train a sentiment analysis model on labelled data .

Use Logistic Regression for modelling the tweets for Sentiment Analysis

Thank You