SPARK PROJECT DOCUMENTATION

lenovo

Overview

- The project involves the implementation of data pipeline that collects Tweets from Twitter on a specific topic which will be on Tesla on our case using the Twitter API.

- The pipeline has **Five Stages**:

    1. Data Source System
    2. Data Collection System
    3. Landing Data Persistence
    4. Landing to Raw ETL
    5. Raw to Processed ETL

- Our goal is to move the data through this pipeline to reach the final stage and make analysis on it.

Technologies Used

- In order to deal with the steps of the pipeline those technologies were included

    - Hadoop (HDFS)
    - Spark
    - Hive
    - Flask

Architecture Details

1- Data Source System:

- Kindly check **steps** file knows the steps of how to run the project.
- We are going to run **step1-run-prerequsites.sh** script it contains the following:
    - Installing the curl
    - Installing Flask and Pyspark
    - Install cron
    - Formation of aliases that will run the project and putting them in bashrc so every time you run the alias it does the same work.
    - Formation of the directories that we are going to work on it on the HDFS and giving the permissions needed.

- We will run in the terminal **run_listener** which is an alias that executes the twitter listener.py script**.** This will open a socket and wait for spark to connect, once connected, a flask application will be started in order to be able to get the data from Twitter API.

- In order to trigger this **twitter_listner.py** a web application was made using Flask which is a popular and lightweight Python web framework used for building web applications, this application was made instead of loop inside the script for two reasons:
    1. It is something that will be called from outside so nothing will affect the script or change in the architecture of it.
    2. If I want to make any change in scheduling interval it will be easier.

- Flask application is working on port 8887 and end point (“/run-5-min-batch”) that will be able to fetch the data from twitter.
- In a NEW terminal run **run_project**

- We will form the concept of scheduling using crontab by adding the following two scripts in the crontab
  1- The first one is cron_file_5_min and this script will run
    - the curl that will initiate the flask application to fetch the data from Twitter
    - the Hivesql scripts that create the landing table and the dimensions and load data into them
  2- The second one is cron_file_1_hour
    - This will run the script that load the data into the fact tables

- To make this crontab we will open a NEW Terminal and write the following:
  - Write the command crontab -e
  - A file will be opened and we will form the scheduling in it:
    - */5 * * * * . ~/itversity-material/cron_file_5_min
    - 0 */1 * * * . ~/itversity-material/cron_file_1_hour

2-Data Collection System:

- In this step after running **pyspark_tweets_sourcin.py** at the end we will have json files having the information that we need from the tweet.
- You will find that I have added a column of batch interval showing the timing that the data will be received in it and that will be used in the analysis.
- You will find that the data is partitioned by Year, Month, day and hour from **created_at** date.

3-Landing Data Persistence:

- In this step we will make hive external table in the script **hive_create_landing.hql** named **spark_tweets_landing** having all the data that we want, partitioned by the previous four columns, stored as parquet and on the HDFS on directory called **landing.**

4 -Landing to Raw ETL:

- In this step we are going to make the dimensions needed, you will find that we have Three dimensions:
    1- **User_dim_raw** having the information about the user
    2- **Tweet_dim_raw** having the information about the tweet
    3- Date_dim_raw having date information from year to hour

- In order to get the data from the landing table to the dimension we needed to think about the updating strategy of the data so I thought that I want it to be **UPSERT to check if the record has any change it will be updated and if it is new it will be inserted and I believe that this will be safer on dealing with data** so I made HIVE TEMPORARY TABLE and this table will contain the data of the last five minutes only and will make MERGE between this table and the dimension table , but I have faced problem doing the Merge that the file type needed to be ORC and that the table need to be transactional so I have solved those problems but in the next step on making the fact from the dimensions unfortunately transactional table was not supported by spark I started to think that I can make view or materialized view but either of them worked you will find this in **landing_to_raw_merge.hql** and therefore I had to it with the append strategy found in **landing_to_raw_no_merge.hql and create_dimension_no_trans.hql.**

5 - Raw to Processed ETL:

- Two simple fact tables were made using spark:
    1- **user_fact_query** for counting the tweets per user
    2- **Tweet_fact_query** for couting the tweets per hours