GEO_SON PROJECT

Problem statement

1.1 Data Collection Module

The Data Collection Module is responsible for:

1. Receive geo-tagged measurement reports or pm events from STEM or ENIQ Event or a geo-location engine. The tool shall receive this information through streaming (or 15min ROP file if streaming is not available.)
2. Receive raw PM events from STEM or ENIQ Event. The delay shall not be longer than 10 seconds for streaming or 15 min for file based reception.
3. Receive PM counter reports from OSS as made available.
4. Pre-process geo-tagged RF measurements to extract *Measurement Report Record.* A *Measurement Report Record* consists of (cell ID, RSRP, RSRQ). Update RF maps with the new set of geo-tagged RF measurements according to section 5 below
5.

    i. The average RSRP per *Bin* location, *RSRP_bin_avg_dBm_new*, is calculated as: *RSRP_bin_avg_dBm_new* = 10*log10((10^(RSRP_1(dBm)/10) + 10^(RSRP_2(dBm)/10) + … + 10^(RSRP_N_new(dBm)/10)) / *N_new*), where *N_new* is the number of *Measurement Report Records* for this cell at this *Bin* location for the e-tilt being considered in the new dataset.

    ii. The average RSRQ per *Bin* location, *RSRQ_bin_avg_dB_new*, is calculated as: *RSRQ_bin_avg_dB_new*= 10*log10((10^(RSRQ_1(dB)/10) + 10^(RSRQ_2(dB)/10) + … + 10^(RSRQ_N_new(dB)/10)) / *N_new*), where *N_new* is the number of *Measurement Report Records* for this cell at this *Bin* location for the e-tilt being considered in the new dataset.

a. Update the existing RF map with the new geo-binned Measurement Report Records, as follows:

    i. *RSRP_bin_avg_dBm_update* = 10*log10(*RSRP_bin_avg_linear_update*), where *RSRP_bin_avg_linear_update* = [*N_old* * 10^(*RSRP_bin_avg_dBm_old*/10) + *N_new* * 10^(*RSRP_bin_avg_dBm_new*/10)]/ (*N_old* + *N_new*). Here, *RSRP_bin_avg_dBm_old* is the current value for the average RSRP and *N_old* is the current number of *Measurement Report Records* for this cell at this *Bin* for the e-tilt being.

    ii. *RSRQ_bin_avg_dB_update* = 10*log10(*RSPQ_bin_avg_linear_update*), where *RSRQ_bin_avg_linear_update* = [*N_old* * 10^(*RSRQ_bin_avg_dB_old*/10) + *N_new* * 10^(*RSRQ_bin_avg_dB_new*/10)]/ (*N_old* + *N_new*). Here, *RSRQ_bin_avg_dB_old* is the current value for the average RSRQ and *N_old* is the current number of *Measurement Report Records* for this cell at this *Bin* for the e-tilt being.

iii. *N_update* = max(*N_old* + *N_new, 10^6)*

Pre-Requisites

1. Considering we have a 2 Node cluster with a Master and Slave.
2. Ensure that Flume agent running on a slave node.(Source for the Flume)
3. Fluming the data onto hdfs to a folder (hdfs ://<master host name>:54310/user/flume).
4. Assuming the input file is ftped to the slave node from the external system every 15 mins.
5. Hadoop, hbase , and mysql should be up and running.
6. Sqoop should be available for the export of the data from hive to mysql.

THRU HIVE IMPLEMENTATION and LOADING into HBASE table.

Step1: Data loading into HIVE tables

1. Data will be available in the local file system folder.
2. Preparation of the DDL statements for the creation of the HIVE Table and hbase table.**(GEO_TAGGED_MEASUREMENTS_DDL_HBASE.sql**).
   Issues :
   1. determination of the correct data type need to be taken care of otherwise we would not get the actual data loaded from the file instead if there is a mismatch of datatype of the column and the data that is loaded will cause NULL's to be inserted.
   2. Make sure all the daemons are running before we execute the above sql. Faced of issues with the HABSE Master not running sometimes because of lot of issues.
   3. We also have to make sure appropriate jars are supplied to HIVE for integration with HBASE.

3. Command to load the .csv file from local onto the HIVE table on HDFS **(GEO_TAGGED_MEASUREMENTS_HBASE.sql).**
   1.Both the 1 and 3 above achieved in a single command by having these commands embedded in a .sql file and using the hive –f option to load the file into the table.
   Inputs: Taken a Smaller .csv file placed in the local path
   2.The sql also contains logic to fetch the required dataset (FINAL AGGREGATED RESULTS 5 section from the requirement document) from the data that is loaded into the HIVE table and loads the same into the HBASE table.

CheckPoint:

1. If the data successfully load into HIVE tables or not. Show tables will list the new table**(GEO_TAGGED_MEASUREMENTS_HIVE)**.

2. If the data successfully load into HIVE table **(HBASE_GEO_TAGGED_MEASUREMENTS_HIVE)** and HBASE table **(GEO_MEASUREMENTS_HBASE)** or not.

Steps to execute the code

1. Copy the below artifacts into the a local folder on the Master Node or Machine
   a. GEO_TAGGED_MEASUREMENTS_DDL_HBASE.sql
   b. GEO_TAGGED_MEASUREMENTS_HBASE.sql
   c. geo_call_data_nohead.csv (removed the first line of the file as it contains column headers).
   d. Execute the below statements from CLI from the same folder where these artifacts are available.

      [hduser@master ~]$ hive -f GEO_TAGGED_MEASUREMENTS_DDL.sql

      (This command will create the necessary tables required (HIVE and HBASE TABLES).

      [hduser@master ~]$ hive -f GEO_TAGGED_MEASUREMENTS.sql

      (This command will query the base hive table where the data gets loaded and performs the necessary calculations and stores the intermediate results into the hive and hbase tables.)

   e. Connect to the hbase and hive shell and check for the results in the tables HBASE_GEO_TAGGED_MEASUREMENTS_HIVE(HIVE table) and GEO_MEASUREMENTS_HBASE(HBASE tables).

Showing these aggregated results from the UI. Used **Tableau** Tool to connect to the HBASE table(Normally no connector available) but there are third party vendors like Simbha technologies who provide a HBASE ODBC connector to connect to the Tableau.

THRU HIVE IMPLEMENTATION and LOADING into MYSQL table.

Step1: Data loading into HIVE tables

1. Data will be available in the local file system folder.
2. Preparation of the DDL statements for the creation of the HIVE Table and hbase table.**(GEO_TAGGED_MEASUREMENTS_DDL_MYSQL.sql**).

Issues :

1. determination of the correct data type need to be taken care of otherwise we would not get the actual data loaded from the file instead if there is a mismatch of datatype of the column and the data that is loaded will cause NULL's to be inserted.

3. Preparation of the DDL statements for the creation of the MYSQL Table .**(GEO_TAGGED_MEAMENTS_MYSQL_DDL.sql**).

4. Command to load the .csv file from local onto the HIVE table on HDFS **(GEO_TAGGED_MEASUREMENTS_MYSQL.sql).**

1.Both the 1 and 4 above achieved in a single command by having these commands embedded in a .sql file and using the hive –f option to load the file into the table.

Inputs: Taken a Smaller .csv file placed in the local path

2.The sql also contains logic to fetch the required dataset (FINAL AGGREGATED RESULTS 5 section from the requirement document) from the data that is loaded into the HIVE table.

CheckPoint:

1. If the data successfully load into HIVE tables or not. Show tables will list the new table**(GEO_TAGGED_MEASUREMENTS_HIVE_MYSQL)**.
2. If the data successfully load into HIVE table (GEO_TAGGED_MEASUREMENTS_HIVE_INTMEDTABLE)

Steps to execute the code

Copy the below artifacts into the a local folder on the Master Node or Machine

a. GEO_TAGGED_MEASUREMENTS_DDL_MYSQL.sql
b. GEO_TAGGED_MEASUREMENTS_MYSQL.sql
c. geo_call_data_nohead.csv (removed the first line of the file as it contains column headers).
d. GEO_TAGGED_MEAMENTS_MYSQL_DDL.sql
e. Execute the below statements from CLI from the same folder where these artifacts are available.

[hduser@master ~]$ hive -f GEO_TAGGED_MEASUREMENTS_DDL_MYSQL.sql
(This command will create the necessary tables required (HIVE)

[hduser@master ~]$ hive -f GEO_TAGGED_MEASUREMENTS_MYSQL.sql

(This command will query the base hive table where the data gets loaded and performs the necessary calculations and stores the intermediate results into the hive table.)

f. Connect to the hive shell and check for the results in the tables

GEO_TAGGED_MEASUREMENTS_HIVE_INTMEDTABLE (HIVE table)

g. Creation of the MYSQL table run the below command.

[hduser@master ~]$ mysql -u root GEO_TAGGED_MEASUREMENTS_MYSQL <GEO_TAGGED_MEAMENTS_MYSQL_DDL.sql

a. Exporting the data from the HIVE table into MYSQL table using the below command.

[hduser@master ~]$ sqoop export --connect jdbc:mysql://master/GEO_TAGGED_MEASUREMENTS_MYSQL --username root --table GEO_TAGGED_MESMNTS_MYSQL --export-dir /user/hive/warehouse/geo_tagged_measurements.db/geo_tagged_measurements_hive_intmedtable/ --input-null-non-string "\\\\N" --input-fields-terminated-by '\0001'

Showing these aggregated results from the UI. Used **iReport-5.5.0** Tool to connect to the MYSQL table.