BDH Assignment

Dalon Lobo

July 30, 2017

Question

Dropped Call Analysis: Lets say this telecom company for which sample data is provided, wants to carry out analytics to find out the most the troublesome Cell (Telecom equipment) and reason for most of the dropping calls.

In order to carry out this analytics you need to perform following steps using Java/Hive or Pig–

- Group CDR Rows from same “sim-card-id” + “Phone number”.
- Compare each row with previous row (from same “sim-card-id” + “Phone number”) and check if there is a new call within less than 1 minute from last call (“start time” + duration). If yes then that call was dropped.
- If yes - compute “cell-id” where call has been dropped and “drop reason”.
- Compute top “cell-id” with dropped calls from the data set and store them into new Hive table “call_drop_details”.

After completing the above steps, answer the following :

Q1: Classify all the call records into 3 categories: ”Dropped“,”Not_Dropped" or “NA”. You can store this output into a Hive table.

Q2: Find top 10 cell ids from which maximum call dropped happened and respective reasons.

Answer

I will use Hive to solve this problem.

Step 1

Setting the hive execution engine to mapreduce

```
# This is used to set the execution engine to mapreduce
SET hive.execution.engine=mr;
```

Step 2

Set hive to use my database i.e. dalon_test.

```
use dalon_test;
```

Step 3

Creating hive table called cdr_data to import the cdr data

```
# Creating the table with all the fields

CREATE EXTERNAL TABLE cdr_data(sessionID string,
                               simcardID string,
                               phoneNumber string,
                               datetime string,
                               duration int,
                               cellID string,
                               networkType int,
                               dropReason string)
COMMENT 'This table contains CDR'
ROW format delimited
fields terminated BY ',' stored AS textfile;

# Describe table details

describe cdr_data;
```

Step 4

Importing the data from hdfs directory to cdr_data table.

```
# Loading the sample.csv into cdr_data table
LOAD DATA inpath '/user/dalonlobo2857/Asmt_CDR' INTO TABLE cdr_data;
```

Verify

Checking the data in cdr_data table

```
# Running select query to check the amount of data in cdr_data table
SELECT count(*) from cdr_data;
```

Output

id	count
0	168

Step 5

Partitioning the table as cdr_data_partitioned on simcardID and loading data into it.

```
# Creating a partitioned table for better data handling

CREATE EXTERNAL TABLE cdr_data_partitioned(sessionID string,
                      phoneNumber string,
                      datetime string,
                      duration int,
                      cellID string,
                      networkType int,
                      dropReason string)
            COMMENT 'This table contains CDR data partitioned on simcardID'
            partitioned BY (simcardID string)
            ROW format delimited
                fields terminated BY ','
            stored AS textfile;

# Loading the data into partitioned cdr table from cdr_data table

INSERT INTO
   TABLE cdr_data_partitioned PARTITION (simcardID)
   select
      sessionID,
      phoneNumber,
      datetime,
      duration,
      cellID,
      networkType,
      dropReason,
      simcardID
   FROM
      cdr_data;
```

Step 6

Creating temporary table cdr_data_tem, where the datetime is converted to unix timestamp and multiplied by 1000 to convert to milliseconds, which will help us in analysing the data easily.

```
# creating temporary table

create table cdr_data_tem (sessionID string,
                           simid string,
                      milliseconds bigint,
                      duration int,
                      cellID string,
                      dropReason string)
                      ROW format delimited
                fields terminated BY ','
            stored AS textfile;

# Inserting into temporary table
INSERT INTO
   TABLE cdr_data_tem
      select sessionID, concat(simcardID,phoneNumber) as simid,
(unix_timestamp(datetime, 'dd/MM/yyyy HH:mm:ss') * 1000) as milliseconds,
duration,
cellID, dropReason
from cdr_data_partitioned ;
```

Step 7

Writing a select query, which can be used categorise the data

```
SELECT a.sessionID,
       a.simid,
       a.cellid,
       a.dropreason,
       (CASE
            WHEN (abs(a.milliseconds - (lag + lagduration))>=60000) THEN 'Not_Dropped'
            WHEN (abs(a.milliseconds - (lag + lagduration))<60000) THEN 'Dropped'
            ELSE 'NA'
        END) AS category,
       a.milliseconds,
       abs(a.milliseconds - (lag + lagduration)) AS d,
       lag,
       lagduration
FROM
  (SELECT sessionID,
          simid,
          cellid,
          dropreason,
          milliseconds,
          duration,
          lag(milliseconds) over (partition BY simid
                             ORDER BY milliseconds,duration ASC) AS lag,
          lag(duration) over (partition BY simid
                              ORDER BY milliseconds,duration ASC) AS lagduration
```

```
   FROM cdr_data_tem) AS a;
```

Output

id	a.sessionid	a.simid	a.cellid	a.dropreason	category	a.seconds	d	lag	lagdura
0	drop-a00…	SIM-00001…	cell H	REASON 4	NA	1430141807000	NULL	NULL	NULL
1	drop-a00…	SIM-00001…	cell H	REASON 4	Dropped	1430141811000	3998	1430141807000	2
2	drop-76.…	SIM-00001…	cell B	REASON 6	NA	1430141814000	NULL	NULL	NULL

more…

Answer to question 1

Step 8

Classify the data into Not_Dropped, Dropped and NA and save in classified_cdr_data table

```
# Creating the table

CREATE TABLE classified_cdr_data (sessionID string,
                                  simid string,
                                  cellID string,
                                  dropReason string,
                                  category string)
ROW format delimited
fields terminated BY ',' stored AS textfile;

# Inserting into the table
INSERT INTO TABLE classified_cdr_data
SELECT a.sessionID,
       a.simid,
       a.cellid,
       a.dropreason,
       (CASE
            WHEN (abs(a.milliseconds - (lag + lagduration))>=60000) THEN 'Not_Dropped'
            WHEN (abs(a.milliseconds - (lag + lagduration))<60000) THEN 'Dropped'
            ELSE 'NA'
        END) AS category
FROM
  (SELECT sessionID,
          simid,
          cellid,
          dropreason,
          milliseconds,
          duration,
          lag(milliseconds) over (partition BY simid
                             ORDER BY milliseconds,duration ASC) AS lag,
          lag(duration) over (partition BY simid
                              ORDER BY milliseconds,duration ASC) AS lagduration
   FROM cdr_data_tem) AS a;
) AS a;
```

```
# Display

select * from classified_cdr_data;
```

Output

id	sessionid	simid	cellid	dropreason	category
0	drop-a00f8eea-5416-4aee-b6a1-e63c8724b352	SIM-00001-20PHONE-00009-20	cell H	REASON 4	NA
1	drop-a00f8eea-5416-4aee-b6a1-e63c8724b352	SIM-00001-20PHONE-00009-20	cell H	REASON 4	Dropped
2	drop-76a06907-ed97-4dd2-9d90-244b14e4e762	SIM-00001-26PHONE-00001-26	cell B	REASON 6	NA

more…

Step 9

Creating call_drop_details table and inserting the data into it.

```
# creating table

CREATE EXTERNAL TABLE call_drop_details ( cellID string,
                                          dropReason string,
                                          dropCount int)
ROW format delimited
fields terminated BY ',' stored AS textfile;

# Inserting the data

INSERT INTO TABLE call_drop_details
SELECT cellID,
       dropReason,
       count(dropReason) AS dropCount
FROM classified_cdr_data
WHERE category = 'Dropped'
GROUP BY cellID,
         dropReason;
```

Answer to question 2

Step 10

Displaying the top 10 cell ids from which maximum call dropped has happened and respective reasons.

```
SELECT cellID,
       dropReason,
       dropCount
FROM call_drop_details
ORDER BY dropCount DESC
LIMIT 10;
```

Output

id	cellid	dropreason	dropcount
0	cell J	REASON 2	3
1	cell H	REASON 10	3
2	cell G	REASON 7	3
3	cell A	REASON 7	3
4	cell G	REASON 2	3
5	cell A	REASON 5	3
6	cell A	REASON 3	2
7	cell B	REASON 10	2
8	cell G	REASON 3	2
9	cell H	REASON 4	2

Step 11

Finding number of dropped calls in the cdr data provided.

```
select sum(dropCount) as count from call_drop_details;
```

Output

id	count
0	81