Big Data on AWS

Lab Three: Data Processing with Hive on EMR

Table of Contents

[Lab Three: Data Processing with Hive on EMR 3](#_Toc509605413)

[Task One: Run Hive interactively 4](#_Toc509605414)

[Task Two: Create tables from log data stored in Amazon S3 5](#_Toc509605415)

[Task Three: Combining the Clicks and Impressions Tables 6](#_Toc509605416)

[Task Four: Use Hive to query the tables in Amazon S3 8](#_Toc509605417)

# Lab Three: Data Processing with Hive on EMR

In this lab, we will use Apache Hive and Amazon EMR to process logs stored in Amazon S3 from servers that serve online advertising. We will process the logs and stored the results in a SQL query-able form in S3.

Through this lab, you will perform the following tasks:

* Task 1: Run Hive interactively
* Task 2: Use Hive commands to create tables from log data stored in Amazon S3
* Task 3: Use Hive commands to join tables and store the joined table in Amazon S3
* Task 4: Use Hive to query the tables in Amazon S3

**Storing logs in S3**

The ad serving machines produce two types of log files: impression logs and click logs. Every time we display an advertisement to a customer, we add an entry to the impression log. Every time a customer clicks on an advertisement, we add an entry to the click log.

Every five minutes the ad serving machines push a log file containing the latest set of logs to Amazon S3. This allows us to produce timely analyses of the logs.

**Data Sources**

{

requestBeginTime: "19191901901",

requestEndTime: "19089012890",

browserCookie: "xFHJK21AS6HLASLHAS",

userCookie: "ajhlasH6JASLHbas8",

searchPhrase: "digital cameras"

adId: "jalhdahu789asashja",

impresssionId: "hjakhlasuhiouasd897asdh",

referrer: "http://cooking.com/recipe?id=10231",

hostname: "ec2-12-12-12-12.ec2.amazonaws.com",

modelId: "asdjhklasd7812hjkasdhl",

processId: "12901",

threadId: "112121",

timers: {

requestTime: "1910121",

modelLookup: "1129101"

}

counters: {

heapSpace: "1010120912012"

}

}

The JSON records are stored one per line in the log file. This makes it easier to import the log files into Hive as they can be read using the default TextFormat. The above example was reformatted to make it easier to read.

These log files are pushed to S3 in 5 minute intervals at the following location:

s3://elasticmapreduce/samples/hive-ads/tables/impressions/dt=$time/$hostname-$time.log

The form "dt=" is required for Hive to be able to import the log files as a partitioned table. Using a partitioned table allows us to specify queries that only read a subset of the data rather than causing Hive to do a full table scan.

The web server boxes also produce click logs which have the following form.

{

requestBeginTime: "19191901901",

requestEndTime: "19089012890",

browserCookie: "xFHJK21AS6HLASLHAS",

userCookie: "ajhlasH6JASLHbas8",

adId: "jalhdahu789asashja",

impresssionId: "hjakhlasuhiouasd897asdh",

clickId: "ashda8ah8asdp1uahipsd",

referrer: "http://recipes.com/",

directedTo: "http://cooking.com/"

}

Like the impression logs, these files are also pushed into Amazon S3 at the following location:

s3://elasticmapreduce/samples/hive-ads/tables/clicks/dt=$time/$hostname-$time.log

## Task One: Run Hive interactively

1. Connect to the Hadoop Master Node.
2. Create logging directory for Hive.

sudo chown hadoop -R /var/log/hive

mkdir /var/log/hive/user

mkdir /var/log/hive/user/hadoop

1. Run the following command to start the Hive shell, change the output-bucket name to your bucket.

hive \

-d SAMPLE=s3://elasticmapreduce/samples/hive-ads \

-d DAY=2009-04-13 -d HOUR=08 \

-d NEXT\_DAY=2009-04-13 -d NEXT\_HOUR=09 \

-d OUTPUT=s3://**<output-bucket>**/samples/output

1. Run the following command and observe that there are no tables in hive metastore.

show tables;

## Task Two: Create tables from log data stored in Amazon S3

Now we shall tell Hive about our impressions data by creating an external table.

1. Create Hive External table with the below command using JSONSerDe.

CREATE EXTERNAL TABLE impressions (

requestBeginTime string,

adId string,

impressionId string,

referrer string,

userAgent string,

userCookie string,

ip string

)

PARTITIONED BY (dt string)

ROW FORMAT

serde 'org.apache.hive.hcatalog.data.JsonSerDe'

with serdeproperties ( 'paths'='requestBeginTime, adId,

impressionId, referrer, userAgent, userCookie, ip' )

LOCATION '${SAMPLE}/tables/impressions' ;

1. The table is partitioned based on time. As yet, Hive doesn't know which partitions exist in the table. We can tell Hive about the existence of a single partition using the following statement.

ALTER TABLE impressions ADD PARTITION (dt='2009-04-13-08-05');

1. If we were to query the table at this point the results would contain data from just this partition. We can instruct Hive to recover all partitions by inspecting the data stored in Amazon S3 using the MSCK REPAIR statement.

MSCK REPAIR TABLE impressions;

1. Follow the same process to recover clicks.

CREATE EXTERNAL TABLE clicks (

impressionId string

)

PARTITIONED BY (dt string)

ROW FORMAT

SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'

WITH SERDEPROPERTIES ( 'paths'='impressionId' )

LOCATION '${SAMPLE}/tables/clicks' ;

MSCK REPAIR TABLE clicks;

## Task Three: Combining the Clicks and Impressions Tables

1. We want to combine the clicks and impressions tables so that we have a record of whether each impression resulted in a click. We'd like this data stored in Amazon S3 so that it can be used as input to other jobs.

CREATE EXTERNAL TABLE joined\_impressions (

requestBeginTime string, adId string, impressionId string, referrer string,

userAgent string, userCookie string, ip string, clicked Boolean

)

PARTITIONED BY (day string, hour string)

STORED AS SEQUENCEFILE

LOCATION '${OUTPUT}/joined\_impressions'

;

This table is partitioned as well. An advantage of partitioning tables stored in Amazon S3 is that if Hive needs only some of the partitions to answer the query then only the data from these partitions will be downloaded from Amazon S3.

The joined\_impressions table is stored in SEQUENCEFILE format, which is a native Hadoop file format that is more compressed and has better performance than JSON files.

1. Next, we create some temporary tables in the local HDFS partition to store intermediate impression and click data.

CREATE TABLE tmp\_impressions (

requestBeginTime string, adId string, impressionId string, referrer string, userAgent string, userCookie string, ip string )

STORED AS SEQUENCEFILE;

1. We insert data from the impressions table for the time duration we're interested in. Note that because the impressions table is partitioned only the relevant partitions will be read.

INSERT OVERWRITE TABLE tmp\_impressions

SELECT

from\_unixtime(cast((cast(i.requestBeginTime as bigint) / 1000) as int)) requestBeginTime, i.adId, i.impressionId, i.referrer, i.userAgent, i.userCookie, i.ip

FROM

impressions i

WHERE

i.dt >= '${DAY}-${HOUR}-00' and i.dt < '${NEXT\_DAY}-${NEXT\_HOUR}-00'

;

1. The start of the time period is DAY-HOUR and the end of the period is NEXT\_DAY-NEXT\_HOUR. NEXT\_DAY is the day of the next time period. It differs from ${DAY} only when we're processing the last hour of a day. In this case the time period ends on the next day.

For clicks, we extend the period of time over which we join by 20 minutes. Meaning we accept a click that occurred up to 20 minutes after the impression.

CREATE TABLE tmp\_clicks (

impressionId string

) STORED AS SEQUENCEFILE;

INSERT OVERWRITE TABLE tmp\_clicks

SELECT

impressionId

FROM

clicks c

WHERE

c.dt >= '${DAY}-${HOUR}-00' AND c.dt < '${NEXT\_DAY}-${NEXT\_HOUR}-20'

;

1. Now we combine the impressions and clicks tables using a left outer join. This way any impressions that did not result in a click are preserved. This join also enables us to search for clicks that occurred after the time period. The query also excludes any clicks that did not originate from an impression in the selected time period.

INSERT OVERWRITE TABLE joined\_impressions PARTITION (day='${DAY}', hour='${HOUR}')

SELECT

i.requestBeginTime, i.adId, i.impressionId, i.referrer, i.userAgent, i.userCookie,

i.ip, (c.impressionId is not null) clicked

FROM

tmp\_impressions i LEFT OUTER JOIN tmp\_clicks c ON i.impressionId = c.impressionId

;

All the above could be run in script mode as well.

## Task Four: Use Hive to query the tables in Amazon S3

1. Observe the output in the output S3 bucket from S3 console.
2. Because the joined\_impressions table is located in Amazon S3 this data is now available for other jobs to use.