Exercise 1

Total Available Points : 70

Exercise Name: End to End Processing of Storage and Retrieval of Data

Subject Name : Information Storage and Retrieval

Week #: 4,5,6,7;

Expected Time Spent per week: 3 to 5 hours

Updated 9/23/15

Table of Content

1. Introduction:

In this Exercise, you will implement end to end storage and retrieval Processes, covering the following features:

- Build a Data Model consisting of transactional data and master data.
- Load data into two types of storage systems : Hadoop and Postgres
- Profile data in order to learn about a dataset that you have just received
- Build aggregate datasets for retrievals or reporting
- Benchmark system/tools for your purposes
- Build Capacity Plan for storing operational and aggregate data

2. Use Case:

In Web analytics, Sessions or visits are the main measures. This can help acquiring new users by finding guest users' behaviors, activating them by finding their product or service interests and promote for continuous activities in the site. This data set has been the heart of many online retail companies.

In this exercise, we will be using Session Log dataset as the main dataset. However, master data like users information will also be used, so various necessary web analytics at user level.

3. Exercise Execution Guideline: Week by Week

Here is an overall guideline of implementing the end to end storage and retrieval processes covering from data modeling, benchmarking and to capacity planning.

Week 4
Step 1: Load local file system data into HDFS
Step 2: Create a Data Model and DFD for Session Log Data and User's Data
Step 3: Create Hive DDL on HDFS

Week 5
Step 4: Data Profiling
Step 5: Read Hive Data and aggregate into another Semantic table
Step 6: Load data from hdfs to Postgres

Week 6
Step 7: Hive query to read core data and semantic data
Step 8: Query data in HDFS using Spark SQL

Week 7

Step 9: Benchmark Hive vs Spark Sql
Step 10: Capacity Plan for 1 year of Web Log data (Postgres, HDFS)

4. Submissions, Timeline, Assessment Criteria :

Submission 1: 15 Points
Submission Week: Week 5
Submission Items:

a. Data Model
b. DFD
c. Hive DDL

Submission 2: 15 Points
Submission Week: Week 6
Submission Items:

a. Data Profiling Results for each column of the Web log data provided.
i.e.

Table Name	Col1	Col2	Col2
Max			
Min			
Count			
Distinct Count			
Common Values			

b. Data Summary report of the Hive Aggregate table. Produce the explain plan for one of the aggregate hive queries that you used earlier.
i.e.

User	Session First Dt	Session Last Dt	Total Duration

Submission 3: 40 Points
Submission Week: Week 7
Submission Items:

a. Capacity Plan (PDF) : Refer to your respective Lab

b. Benchmark Results : Comparison of the query performance you saw using SparkSQL vs. Hive along with a graph (bar)
c. Aggregate Postgres table DDL and results (PDF) : you can submit count of rows in that table, the User_id who used the max no. of sessions.
d. Overall Project Execution Log (PDF) : For each step, that you would do, you will require to take log of your activities for each step and submit at the end. None of the steps should be missed.

8. Infrastructure:

Amazon EC2, AMI, S3, Github
You will be using the Amazon EC2 student's account which is provided to you by UCB. You will be accessing the AMI provided as well to create your own server and work on this.

You can use either of these two AMI's:

UCB W205 Base - ami-98848cf0 : it's the same AMI used for your labs.

ucb_w205_complete - ami-71cdb014 : It is a different AMI with separate Hadoop processed. Here are some instructions for what you need to do:

- A user, w205, under which jobs should be run
- Default configuration for HDFS, YARN, Hive w/Derby and Spark 1.5
- 2 scripts start-hadoop.sh and stop-hadoop.sh for starting and stopping Hadoop services

To use the AMI, you need to do the following. It will not work otherwise:

- Attach storage (preferably an EBS volume) to the instance
- Mount that storage under /data
- Make sure formatted NameNode data is created on /data
  - This means the first time you run the instance with a new mounted volume, you'll need to run:
    - sudo -u hdfs hdfs namenode -format
    - start-hadoop.sh

9. Technology :

Python, HDFS, Postgres, Hive, Apache Spark, Apache Sqoop, Amazon EC2, Amazon S3, Cloudera Express, github

10. Data Set:

Web Log Data (Simulated)
User Data (Master Data: Simulated)

The Datasets are hosted at:
https://s3.amazonaws.com/ucbdatasciencew205/exercise_1/userdata.csv
https://s3.amazonaws.com/ucbdatasciencew205/exercise_1/weblog.csv

11. Overall Guideline - for all steps

Here is the detail guideline for each of the steps for implementation. You would use the same Amazon AMI for creating your own EC2 server for this exercise. You must have a github account if you wish to store your scripts, data, etc., which is recommended. You may not want to keep your EC2 server live all the time as you will run out of credit that way. So, you could save your work in github as you progress and when you make your sever alive, you can re -pull the code and use.

Week 4
Step 1: S3 to local file system to HDFS

Once you have access to Amazon EC2 Account, please check the above public S3 buckets mentioned above, where the dataset is stored for your exercise.

Now, you login to your server and choose a local directory to load the data from S3 buckets. You could use wget command for doing the same.

i.e. `wget http://s3.amazonaws.com/BucketName/Folder/File_name`

After getting the datafiles from S3 to your local folder on your server, you will load this same data set into HDFS. Here are a few sample commands for the same.

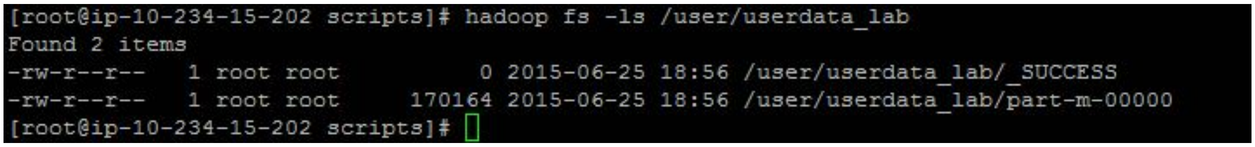
```
hadoop fs -mkdir /user/hadoop/Folder
hadoop fs -mkdir /user/hadoop/Folder/input


hadoop fs -copyFromLocal LocalFileName /user/hadoop/Folder/input
```

To use other HDFS commands, you can use the Help command:

```
hadoop fs -help
```

Make sure you got both the data files into Local File system and HDFS. Note down the sizes and count of records in each of these files.

Here is an example log:



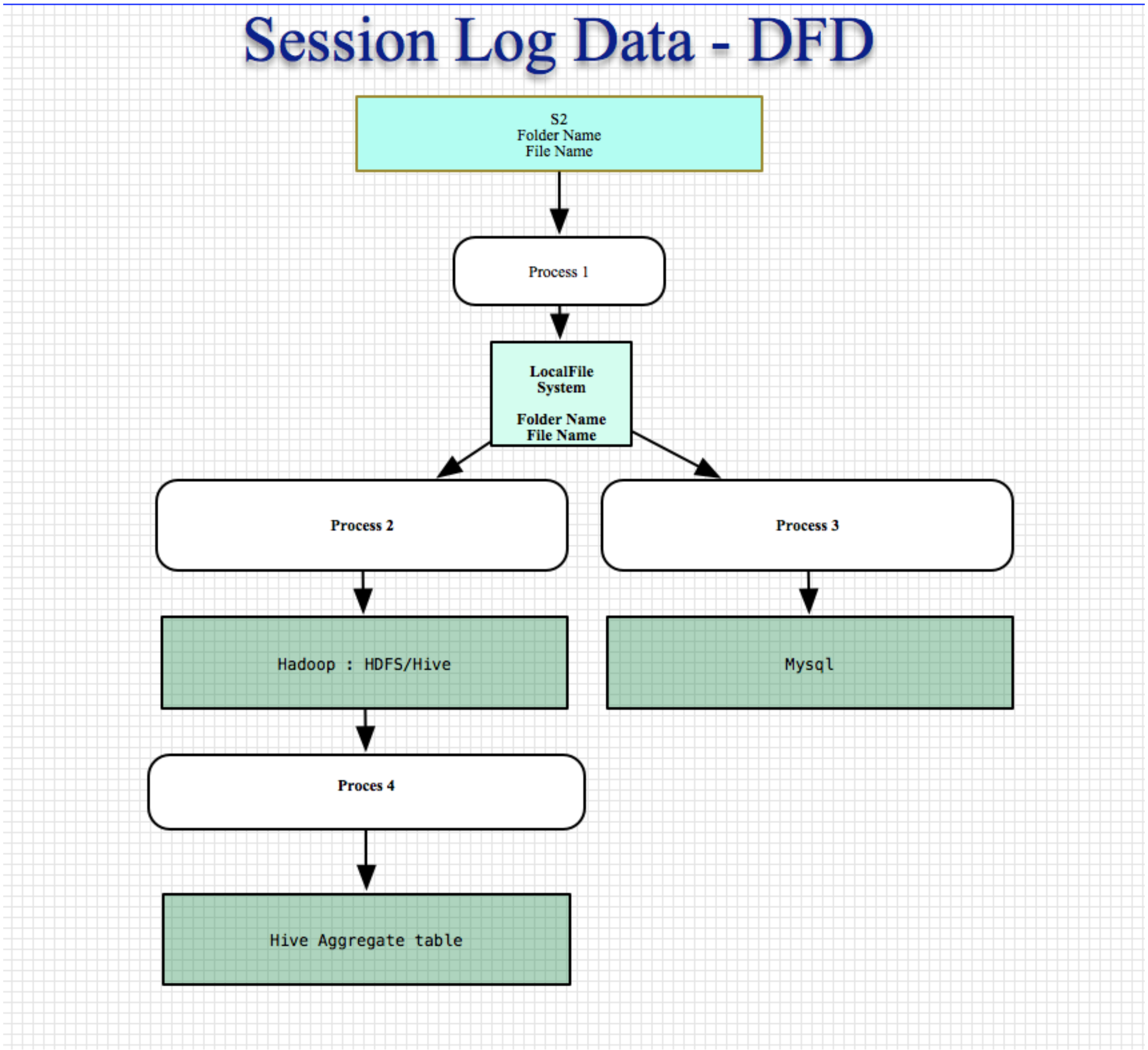
```
[root@ip-10-234-15-202 scripts]# hadoop fs -ls /user/userdata_lab
Found 2 items
-rw-r--r--   1 root root          0 2015-06-25 18:56 /user/userdata_lab/_SUCCESS
-rw-r--r--   1 root root     170164 2015-06-25 18:56 /user/userdata_lab/part-m-00000
[root@ip-10-234-15-202 scripts]# 
```

Deliverables:

1. Screenshot of your files loaded into your hdfs folder
2. Record count of each of these files

Step 2 : Create a Data Model and DFD for Web Log Data and User Data and the Flow

Here is a sample data flow diagram for your reference:



You will need to plot your diagram as a design diagram for your storage and retrieval system for overall data flow. If you have access to any data modeling tools like Erwin, System Architect, you can use. You could use draw.io or any other open source tools of your choice.

Please note that, this is to give a guideline only. You can add as many details like file name, folder name, server name, table name so that this diagram becomes your landscape diagram of the system you are setting up.

Other than this DFD, please create one ER(Entity Relationship) diagram as well at logical and physical level.

Deliverables:

1. Overall Data Flow Diagram including processes, systems, etc.

2. ER Diagram

Step 3: Create Hive DDL on HDFS
Once you have data which you pulled from S3 bucket to HDFS, to query the data, you could create Hive DDL or metadata layer on top of the file system.

You could create the Hive DDLs using the following hive commands as well. Please modify as per your need.

```
CREATE DATABASE IF NOT EXISTS exercise;

use exercise; - this is the schema name; you could change with your own name

CREATE EXTERNAL TABLE IF NOT EXISTS labs.weblog_lab(DATE TIMESTAMP,USERID
STRING,SESSIONID STRING,PRODUCTID STRING,REFERERURL STRING)
 COMMENT 'WEBLOG TABLE'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
 STORED AS TEXTFILE LOCATION '/weblog' tblproperties ("skip.header.line.count"="1");


CREATE EXTERNAL TABLE IF NOT EXISTS userdata_lab(date TIMESTAMP,userid
STRING,firstname STRING,lastname STRING,location STRING)COMMENT 'USERDATA
TABLE'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
 STORED AS TEXTFILE LOCATION '/userdata' tblproperties ("skip.header.line.count"="1");
```

You could also create a script to execute this kind of DDL steps. This will help to log any activity that you conduct in a production like environment.

Please note the format of the file you just created. You must read about various kinds of datafile formats for storing data. Those are RC, ORC, Parquet, etc. with various kinds of compression types. This optimizes your storage and retrieval processes when you design these right.

Now, create two managed tables with parquet file format and select insert into these new tables using data in the previous tables.

Deliverables:

1. Two hive table ddls
2. Two managed table ddls with parquet file format
3. List the datafiles for your hive managed tables with parquet file format

Week 5
Step 4 : Data Profiling

Data profiling is a tactic to learn about your data, which enables you to design better systems, data models, implement Data Quality management systems and storage systems.

In hadoop, you could create your hive queries to profile the data. Usually we find out max, min, total_count, null count, etc. as part of profiling for each of the columns in a hive table.

You need to create your own HQL scripts for your profiling purpose which should be called by a shell or python scripts to store the result data set into a file rather than the screen.

Deliverables:

3. Stats of each column of both the tables you created so far
4. Your data profiling queries as filename.hql

Step 5: Read Hive Data and aggregate into another Semantic table
For this, you could create a summary table based on your dataset provided. Create the Hive DDL and deploy in Hive. You could create HiveDML script for the aggregation query and load to your target Aggregate or Semantic table.

Or, you could simply create this way as an adhoc basis:

```
create table summary table as select u.userid,u.firstname,u.location,w.refererurl from
userdata_lab u,weblog_lab w where u.userid = w.userid ;
```

Or, you could create a wrapper script to execute your Summary Query:

```
#!/bin/sh
if [ "$#" -ne 2 ]; then
  echo "Usage: $0 summaryquery.sql LogFile.txt"
  exit 1
fi

hive -f $1 >> $2
```

Deliverables:

1. Hive ddl for aggregate table
2. Hive DML for aggregating data for weblog and user dataset

Step 6: Load data from hdfs to Postgres

In this step, you will move data from hdfs to Postgres. So, the summary data is moved back to RDBMs for connecting to Reporting tools like Tableau, Microstrategy, etc.

For this you could use Apache Sqoop.

Please refer to Postgres reference link to get familiar with the DB. Here are a few useful commands:

```
starting postgresql:
=================
service postgresql initdb

sudo /etc/init.d/postgresql start

su postgres

create password : alter user postgres PASSWORD 'password';

create database : create database labs;

create role : CREATE ROLE root LOGIN password 'password';

change user to superuser : alter user root with SUPERUSER ;


CREATE DATABASE exercise1;

open psql shell: psql dbname username

\l – show databases;

\d – show tables;

\d table – show clounms

version of psql : select version();
```

Here is another script example of creating a db:

```
    #!/bin/sh
```

```
DB_NAME=$1
if [ "$#" -ne 1 ]; then
echo "provide 1 arg : database name : example -lab"
exit 1
fi
if psql ${DB_NAME} -c '\q' 2>&1; then
   echo "database ${DB_NAME} exists"
else
psql postgres postgres << EOF
create database $DB_NAME;
EOF
fi
```

Here is another script example of creating a Postgres table:

```
#!/bin/sh
username=$1
db_name=$2
if [ "$#" -ne 2 ]; then
  echo "provide 2 args --username and database name : example - root lab"
  exit 1
fi

psql $db_name $username << EOF
CREATE TABLE weblog_lab(date VARCHAR(255),USERID
VARCHAR(255),SESSIONID VARCHAR(255),PRODUCTID
VARCHAR(255),REFERERURL VARCHAR(255));
CREATE TABLE userdata_lab(date VARCHAR(255),userid
VARCHAR(255),firstname VARCHAR(255),lastname VARCHAR(255),location
varchar(255));
CREATE TABLE summarytable(userid VARCHAR(255),firstname
VARCHAR(255),location varchar(255),REFERERURL VARCHAR(255));
EOF
```

You could export the dataset from HDFS to Postgres using below example command.

```
sqoop export --connect jdbc:postgresql://localhost:5432/labs --table
summarytable  --export-dir /user/hive/warehouse/labs.db/summarytable --
username root --password password -m 1 --input-fields-terminated-by ','  ;
```

Deliverables:

1. Reconciliation report of postgres table vs hive aggregate table
2. Postgres table ddl

Week 6

Step 7 : Hive query to read core data and semantic data

These are the queries you need to build to query your Web log and the summary data. For this you could create hql files and execute via a wrapper script.

Deliverables:

1. Hive query for core data
2. Hive query for aggregate data

Step 8 : Query data in HDFS using Spark SQL

As hive is used for batch processing, if you need access to your data on an adhoc basis, Spark SQL is an option for it. You can use “spark-shell” to go to spark prompt, which is “scala>” as well.

For this please refer to your Apache Spark Lab as well if you are new to Spark.

Here is a sample code :

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Running SQL Queries Programmatically :

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.sql(“SELECT * FROM Web_Session_Log limit 10”)
```

Programmatically Specifying the Schema :

```
from pyspark.sql import *
sqlContext = SQLContext(sc)
lines = sc.textFile(“/mnt/weblog.csv”)
parts = lines.map(lambda l: l.split(“\t”))
Web_Session_Log = parts.map(lambda p: (p[0], p[1],p[2], p[3],p[4]))
schemaString = “DATETIME USERID SESSIONID PRODUCTID REFERERURL”
fields = [StructField(field_name, StringType(), True) for field_name in schemaString.split()]
schema = StructType(fields)
schemaPeople = sqlContext.createDataFrame(Web_Session_Log, schema)
schemaPeople.registerTempTable(“Web_Session_Log”)
results = sqlContext.sql(“SELECT USERID FROM Web_Session_Log”)
names = results.map(lambda p: “Name: ” + p.USERID)
for name in names.collect():
print name
```

Deliverables:

1. Spark sql query

2. Spark shell commands to query
3. PySpark execution log

Week 7
Step 9 : Benchmark Hive vs Spark SQL
This step is very important when you are about to chose a product for your purpose. This benchmarking is suitable for when you are choosing a software for adhoc access as well.

Basically, you can create your Hive query with a desired file format in the DDL and same dataset, access via Spark SQL.

Collect and Compare the results.

Deliverables:

1. Query with dataset in Hive
2. Query with dataset in hdfs using Spark-sql
3. Query with dataset in Hive+Parquet
4. Query with dataset in Spark-sql+Parquet
5. Query performance run time for #1, #2, #3, #4

Step 10 : Capacity Plan for 1 year of Web Log data (Postgres, HDFS)
For this, you could use http://www.thecloudcalculator.com/calculators/disk-raid-and-iops.html
as referred in your lab.

Please produce your capacity plan and requirement for Postgres and hdfs with 1 Petabytes of data with daily volume of 10G. The above tool might give you option to plot for only 3G of hard disk space. So, you will need to extrapolate for getting the numbers for 1 Petabyes volume.

Deliverables:

1. For 250 GB volume, cost analysis cloud vs. own
2. For 250 GB volume, project required storage and bandwidth for backup with 2 hours of window per week.

Note:

1. You also need to build your process to load the same dataset to Hadoop/HDFS. You will be building a process very simple to load data. However, this could be a starting building block for a system with even Petabytes of data. When you will load complex dataset, only number of these building blocks will increase, not the complexities much. It is a best

practice to keep a storage and retrieval system simple and manageable as the focus is always in the data and the business value we add from it.

2. Here is a basic syntax of hadoop distcp.
hadoop distcp s3-src hdfs-destination.
Here is an example,
“hadoop distcp
s3n://accesskey:secretekey@bucketname/WebLog_data.txt hdfs://locahost:8020/data/”

   This is very useful for copying data from one hadoop system to another or pull from S3 as an example.

12. Recommended Readings:

1) Postgres : http://www.postgresql.org
2) Apache Sqoop : http://sqoop.apache.org
3) Apache Hive : Hive_DML, Hive DDL, Python Client for JDBC Connection
4) Amazon Web Services : Amazon Web Services
5) Hadoop : Hadoop Tutorials
6) Apache Spark :http://spark.apache.org/docs/1.3.0/