Exercise 1

End to End Processing

of

Storage and Retrieval of Data

ToC

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 transaction data and master data.
- Load data into two types of storage systems : Hadoop and Mysql
- Profile data in order to learn about a dataset that you have just received
- Build aggregate datasets for retrievals or reporting
- Benchmark DB systems 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 3
Step 1 : Load S3/local file system data into Mysql
Step 2 : Load data set from Mysql to hdfs
Step 3 : Create a Data Model and DFD for Session Log Data and User's Data

Week 4
Step 4 : Data Profiling
Step 5 : Hive DDL on HDFS
Step 6 : Read Hive Data and aggregate into another Semantic table
Step 7 : Load data from hdfs to Mysql

Week 5
Step 8 : Hive query to read core data and semantic data
Step 9 : Query raw data in HDFS using Spark SQL

Week 6

Step 9 : Benchmark Hive vs Spark scala
Step 10 : Capacity Plan for 1 year of Session Log data (Mysql, HDFS)

4. Reference Code:

The followings are the code for your reference only. We suggest you to write your code from scratch and only reference when needed.

Revise this list as per current code list.

1. LF_ToMysql.sh
   This code is to load data from your Local File System to Mysql
2. Mysql_ddl.sh
3. MysqlSqoopToHdfs.py
   a. This code is to load data from Mysql to HDFS
4. profiling.sh
   a. This script is a base wrapper script for profiling your dataset
5. HiveDDL.sql
   a. This is one sample Hive DDL script
6. HiveDDL.sh
   a. This is a wrapper script to deploy your DDL.
7. Summary.sh
   a. This script could be used to execute your aggregate processes
8. Summary.hql
   a. This one contains sample Aggregate script
9. sqoop_export.sh
   a. This script is for moving data back to Mysql. You could use for moving summary data back to Mysql
10. SparkSQL.py
    a. This is script for Spark SQL

6. Submissions, Timeline :

Submission 1 :
Submission Week : Week 4
Submission Items :

a. Data Profiling Results for each column of the session log data provided. Format must be in PDF form.
b. Data Summary report of the Hive Aggregate table. Format must be in PDF form.

Submission 2 :
Submission Week : Week 7
Submission Items :

c. Capacity Plan (PDF)
d. Benchmark Results (PDF)
e. Aggregate Mysql table DDL and results (PDF)

7. Assessment Criteria:

Based on Deliverables in Week 4 and Week 7. Please see below the total score per submission item:

Submission Week : Week 4
Submission Items :

f. Data Profiling Results for each column of the session log table (10pts)
g. Data Summary report of the Hive Aggregate table (10pts)

Submission 2 :
Submission Week : Week 7
Submission Items :

h. Capacity Plan (10pts)
i. Benchmark Results (10pts)
j. Aggregate Mysql table DDL and results PDF (10pts)

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.

Here is the AMI Name : **UCB W205 Base** - ami-98848cf0

You can find the reference code in github. You can create your github account if you do not have one already.

The Github Repository name is https://github.com/orgs/UC-Berkeley-I-School/teams/data-science-w205

9. Technology :

Python 2.7.3, HDFS, Mysql 5.1.73, Hive, Apache Spark, Apache Sqoop, Mysql Loader, Amazon EC2, Amazon S3, Cloudera Express 5.4.1, github, S3cmd-1.5.1.2

10. Data Set:

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

You will find the dataset in https://github.com/orgs/UC-Berkeley-I-School/teams/data-science-w205 under exercise 1 folder.

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 to access the UCB github repository in which the reference scripts and data are stored.

Week 3
Step 1 : S3/local file system to Mysql

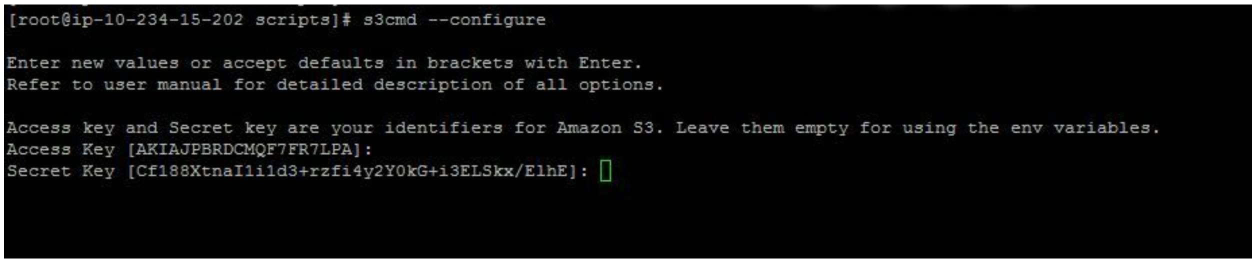
Once you have access to Github, please load the data files to your EC2 Server local folder as per your designed folder structure if any, from the git repository provided above.

You could use the following high level steps for this. However, please get familiar with this in github.com if needed.

After getting the datafiles from git to your local folder, you will load this same data set to S3. Here is a sample command for the same.

```
S3cmd put <local files> <s3://bucketname/destination folder
```

However, you will need to configure S3cmd with your Amazon S3 credentials. Here is a reference for the same.



```
[root@ip-10-234-15-202 scripts]# s3cmd --configure

Enter new values or accept defaults in brackets with Enter.
Refer to user manual for detailed description of all options.

Access key and Secret key are your identifiers for Amazon S3. Leave them empty for using the env variables.
Access Key [AKIAJPBRDCMQF7FR7LPA]:
Secret Key [Cf188XtnaI1i1d3+rzfi4y2Y0kG+i3ELSkx/ElhE]:
```

Next, please write a process to load this dataset to the mysql database available in your server.

You should have these files in your local server to do the next steps.

```
[root@ip-10-234-15-202 exercise]# ls
userdata.csv  weblog.csv
```

You can connect to mysql using the following command :

```
[root@ip-10-234-15-202 exercise]# mysql -u root -p
Enter password:
```

You could follow the following steps for this :

- Connect to Mysql DB
- Create a schema for you project
- Create the tables based on your two datasets.
- Write a python/shell script which will read the datafiles in the local directory and load into the mysql tables you just created and load
- Verify data

Your reference script for this is LF_ToMysql.sh load data to the mysql tables. This script is only a base script for you. You modify as per your need.

You could refer to mysql_ddl.sh for creating mysql tables as reference.

Here is an example of code in python as well.

Pseudo code for Python wrapper script to load data file into MySQL table:

```
import MySQLdb
import os
import string
open database connection
db = MySQLdb.connect (host=”localhost”, port=3307, user=”root”, passwd=”root”, db=”shopgirl”, local_infile = 1)
cursor=db.cursor()
Query Table
sql = “LOAD DATA LOCAL INFILE ‘/home/sany/Downloads/weblog.csv’ INTO TABLE WebLog FIELDS
TERMINATED BY ‘\t’ (DATETIME, USERID, SESSIONID, PRODUCTID, REFERERURL);”
Execute the SQL command:
cursor.execute(sql,row)
#Commit your changes in the database
db.commit()
```

```
#close DB connection
 db.close()
```

After this step, you should have two tables with records in mysql DB and in your respective schema. Please run queries to see how many records are in those tables.

You can directly load to mysql from S3 as well.

Step 2 : Load data set from Mysql to hdfs
In this step, you will load data from Mysql to hdfs, which is a very realistic scenario as OLTP data is usually moved to reporting Systems for analytic purposes.

Please verify your Hadoop installation. You can create a respective hdfs folder into which, you will move data from Mysql to hdfs using Apache sqoop.

You can refer to script MysqlSqoopToHdfs.py for this. You could expand that script to enhance further with your design.

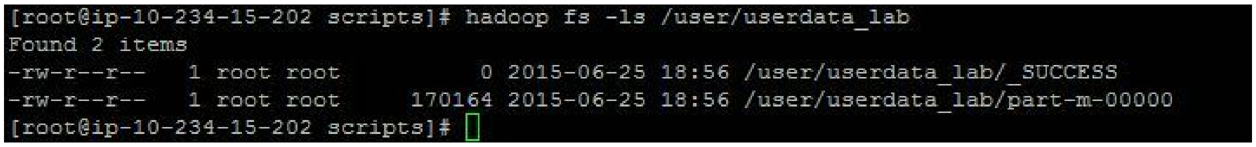
Or you could load this using CLI as follows:

```
sqoop import --connect jdbc:mysql://localhost/labs --username root --password password --table
userdata_lab -m 1 --target-dir  /user/userdata_lab
```

See, I am using only one mapper in the above step.

We recommend you to do this using scripts as in production systems, for automations, you will always need tactics like scripts.

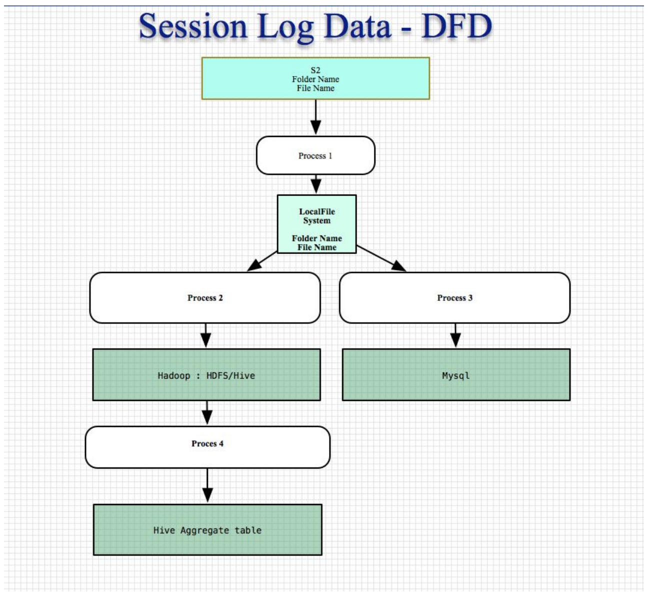
Please verify that the data is loaded to hdfs for both the datasets.



```
[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]#
```

Step 3 : Create a Data Model and DFD for Session 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 part of your storage and retrieval system. If you have access to any data modeling tools like Erwin, System Architect, it is great. However, you could plot real life Data Flow Diagrams using Powerpoint or Keynote as well.

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.



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.

You could use the above script as reference for profiling, Profiling.sh.

You will create your own DDL scripts for your profiling purpose which could be called by a similar script as above.

Step 5 : Hive DDL on HDFS

Once you have data which you pulled from Mysql to HDFS, to query the data, you could create Hive DDL or metadata layer on top of it.

You can refer to script(DDL) for this, HiveDDL.sql.

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

```
CREATE DATABASE IF NOT EXISTS labs;

use labs; - 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. Here is a reference script for the same as HiveDDL.sh

Please note the format of the file you just created. You must read 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 when you design these right.

Step 6 : 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
```

Or, you could refer to Summary.sh and Summary.hql for this as reference. You could expand those scripts.

Step 7 : Load data from hdfs to Mysql

In this step, you will move data from hdfs to mysql. Usually 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 the sqoop_export.sh for the same.

Week 5
Step 8 : Hive query to read core data and semantic data
These are the queries you need to build to query your session log and the summary data. For this you could create hql files and execute via a wrapper script.

Step 9 : Query raw data in HDFS using Spark
As hive is used for batch processing, if you need access to your data on an adhoc basis, Spark is an option for it. You can use “spark-shell” to go to spark prompt, which is “scala>”.

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
```

This could be found in your script list as SparkSql.py or sparksql.sh for your reference.

Week 6
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.

Step 10 : Capacity Plan for 1 year of Session Log data (Mysql, 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 mysql 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.

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) Mysql : http://dev.mysql.com/doc/refman/5.6/en/
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 : https://en.wikipedia.org/wiki/Apache_Spark