**Big Data Management - CS585 - Project 2**

* **Azharuddin Priyotomo**
* **Congyuan Tang**

We are working under the virtual box (Ubuntu OS) and hadoop environment that the Professor provided us.

**1. Query 1**

To perform the query 1, we only need the data set from transaction file, since all output (Customer ID, Number of Transactions and Total Sum) are all available in transaction data.

First we need to load the transaction data into one variable and then group it by customer ID. From the grouped data, we select aggregate count of the transaction and the sum of transaction, and then store it to an output file in HDFS.

Query:

SET DEFAULT\_PARALLEL 20;

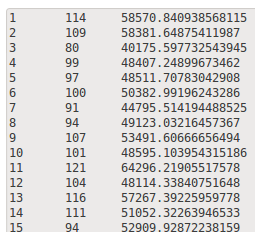
trans = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/transactioninput' USING PigStorage(',') as (transid:int,custid:int, trxamt:float,noitem:int,desc:chararray);

A = group trans by custid;

B = foreach A generate group, COUNT(trans) as NumTransactions, SUM(trans.trxamt) as TotalSum;

STORE B INTO '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/project2/query1output' USING PigStorage();

Sample Output:



Description: (Customer ID, Number of Transactions, Total Sum)

**2. Query 2**

To execute the file for the second query:

1. Start Hadoop and Pig: > start-all.sh

2. Clear output folder for query's result:

> hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp

> hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final

3. Be sure to change the path of files (customer and transaction dataset) in the query2.pig file.

4. Run script in batch mode: > pig PATH/query2.pig

5. The final result will be in folder /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final

The script file for this query is below:

%declare dataSource1 '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Customer'

%declare dataSource2 '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction'

%declare tempOutputFile '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp'

%declare finalOutputFile '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final'

raw1 = LOAD '$dataSource1' USING PigStorage(',') AS (CID1, CName, Age, CountryCode, Salary);

raw2 = LOAD '$dataSource2' USING PigStorage(',') AS (TID, CID2, TransTotal, TransNumItem:INT, TransDesc);

sub\_raw1 = FOREACH raw1 GENERATE CID1, CName, Salary;

joinedTable = JOIN sub\_raw1 BY CID1, raw2 BY CID2;

groupedTable = GROUP joinedTable BY CID1;

temp = FOREACH groupedTable GENERATE group, COUNT(joinedTable.CID1) AS NumOfTrans, SUM(joinedTable.TransTotal) AS TotalSum, MIN(joinedTable.TransNumItem) AS MinItems;

outputTable = JOIN temp BY group, sub\_raw1 BY CID1;

STORE outputTable INTO '$tempOutputFile' USING PigStorage(',');

raw3 = LOAD '$tempOutputFile' USING PigStorage(',') AS (CID, NumOfTrans, TotalSum, MinItems, CID1, CName, Salary);

outputTable2 = FOREACH raw3 GENERATE CID, CName, Salary, NumOfTrans, TotalSum, MinItems;

STORE outputTable2 INTO '$finalOutputFile' USING PigStorage(',');

The query’s result is attached as file ‘query2result.txt’.

The system output (statistics part) is attached as file ‘query2SystemOutput.txt’.

**3. Query 3**

To do query 3, here is the process:

1. We should use two data set (customer and transaction), so we should first join those two data into one variable based on Customer ID.

2. After we have joined data, we group them based on Customer ID and Country Code and aggregate transaction amount using SUM, resulting output of [Customer ID, Country Code, Total Transaction].

3. We perform another separated query of grouping customer data by Country Code and outputting them in format of [Country Code, Total Customer].

4. We join query result from no 3 and no 2 by Country Code. We then group them by Country Code and Total Customer, so we can get the aggregate function of MAX and MIN of the total transaction amount and resulting the final output of [Country Code, Total Customer, Maximum Total Transaction, Minimum Total Transaction].

5. Store the result into file in HDFS.

Query:

SET DEFAULT\_PARALLEL 500;

trans = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/transactioninput' USING PigStorage(',') as (transid:int,custid:int, trxamt:float,noitem:int,desc:chararray);

cust = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/customerinput' USING PigStorage(',') as (id:int,name:chararray, age:int,cc:int,salary:float);

A = join cust by id, trans by custid;

AA = group A by ($0,$3);

AAA = foreach AA generate FLATTEN(group) as (custid, cc), SUM(A.trxamt) as trx;

B = group cust by cc;

BB = foreach B generate group as cc, COUNT(cust.id) as totalcust;

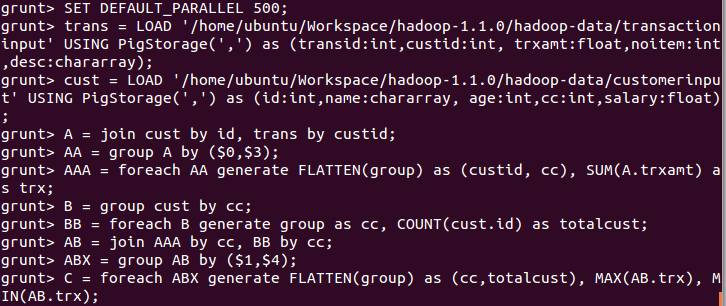
AB = join AAA by cc, BB by cc;

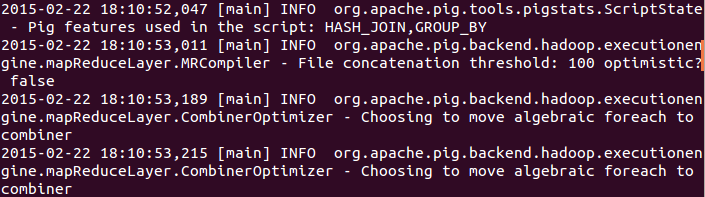
ABX = group AB by ($1,$4);

C = foreach ABX generate FLATTEN(group) as (cc,totalcust), MIN(AB.trx) , MAX(AB.trx);

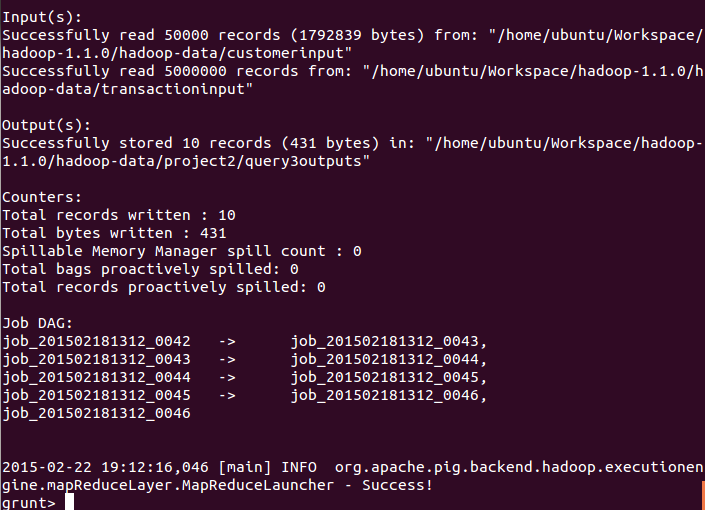
STORE C INTO '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/project2/query3outputs' USING PigStorage();

Screen captures:

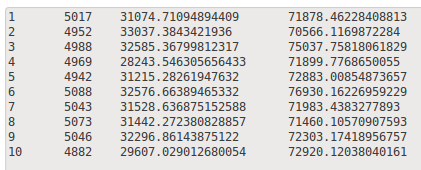




....



Output:



Description: Country Code, Number of Customers, Minimum Total Transaction, Maximum Total Transaction

**4. Query 4**

The process of execution of this query is the same as query2.

The script for this query is below:

%declare dataSource1 '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Customer'

%declare dataSource2 '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction'

%declare tempOutputFile '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-temp'

%declare finalOutputFile '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-final'

raw1 = LOAD '$dataSource1' USING PigStorage(',') AS (CID1, CName, Age, CountryCode, Salary);

raw2 = LOAD '$dataSource2' USING PigStorage(',') AS (TID, CID2, TransTotal, TransNumItem:INT, TransDesc);

sub\_raw1 = FOREACH raw1 GENERATE CID1, CName, Salary;

joinedTable = JOIN raw2 BY CID2, sub\_raw1 BY CID1 USING 'replicated';

groupedTable = GROUP joinedTable BY CID1;

temp = FOREACH groupedTable GENERATE group, COUNT(joinedTable.CID1) AS NumOfTrans, SUM(joinedTable.TransTotal) AS TotalSum, MIN(joinedTable.TransNumItem) AS MinItems;

outputTable = JOIN temp BY group, sub\_raw1 BY CID1 USING 'replicated';

STORE outputTable INTO '$tempOutputFile' USING PigStorage(',');

raw3 = LOAD '$tempOutputFile' USING PigStorage(',') AS (CID, NumOfTrans, TotalSum, MinItems, CID1, CName, Salary);

outputTable2 = FOREACH raw3 GENERATE CID, CName, Salary, NumOfTrans, TotalSum, MinItems;

STORE outputTable2 INTO '$finalOutputFile' USING PigStorage(',');

The query’s result is attached as file ‘query4result.txt’.

The system output (statistics part) is attached as file ‘query4SystemOutput.txt’.

The statistics from query2 is:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features

1.1.0 0.10.0 ubuntu 2015-02-22 04:55:24 2015-02-22 04:58:35 HASH\_JOIN,GROUP\_BY

Success!

Job Stats (time in seconds):

JobId Maps Reduces MaxMapTime MinMapTIme AvgMapTime MaxReduceTime MinReduceTime AvgReduceTime Alias Feature Outputs

job\_201502220216\_0030 1 0 2 2 2 0 0 0raw1,sub\_raw1 MAP\_ONLY

job\_201502220216\_0031 6 1 10 0 7 57 57 57 joinedTable,raw2 HASH\_JOIN

job\_201502220216\_0032 8 1 7 3 5 41 41 41 groupedTable,temp GROUP\_BY,COMBINER

job\_201502220216\_0033 2 1 1 1 1 10 10 10 outputTable HASH\_JOIN /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp,

job\_201502220216\_0034 1 0 2 2 2 0 0 0outputTable2,raw3 MAP\_ONLY /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final,

The statistics from query4 is:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features

1.1.0 0.10.0 ubuntu 2015-02-22 05:33:57 2015-02-22 05:36:29 REPLICATED\_JOIN,GROUP\_BY

Success!

Job Stats (time in seconds):

JobId Maps Reduces MaxMapTime MinMapTIme AvgMapTime MaxReduceTime MinReduceTime AvgReduceTime Alias Feature Outputs

job\_201502220216\_0053 1 0 2 2 2 0 0 0raw1,sub\_raw1 MULTI\_QUERY,MAP\_ONLY

job\_201502220216\_0054 5 1 26 12 20 86 86 86 groupedTable,joinedTable,outputTable,raw2,temp REPLICATED\_JOIN,GROUP\_BY,COMBINER /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-temp,

job\_201502220216\_0055 1 0 2 2 2 0 0 0outputTable2,raw3 MAP\_ONLY /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-final,

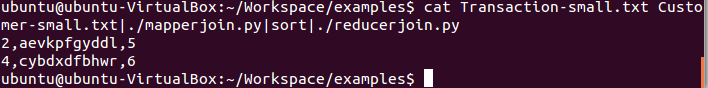
Compared the system output of this query and query2, we can clearly see query2 consists of 5 Map and/or MapReduce jobs, whereas query4 requires 3 Map and/or MapReduce jobs. Also, the time cost for query2 & 4 is 3mins 11secs and 2mins 42secs respectively. Using replicated JOIN increase the efficiency by nearly 15%.

**5. Query 5**

We choose Python language combined with the Hadoop Streaming. To do the query, we need one mapper (mapperjoin.py) and one reducer (reducerjoin.py), where the joining data set will happen in reducer side. The mapper will read the input from both of transaction and customer file. Initial value of (-1) is set for each of the attribute that we need (customer \_id, customer\_name, country\_code, transaction) and will be replaced with real value after reading the input file. So for example, when mapper read data from customer file, the value of transaction\_id will still be -1 while other attributes will have the real values. The output from the mapper will follow the format of [customer\_id, customer\_name, country\_code, transaction\_id]. The output will be sorted and then sent to reducer. Reducer will group the output from the mapper based on **customer ID**, filter the result using condition (if country code = 5) and then sent to stdout.

We first test the codes using the local Python environment first using a small dataset:

cat Transaction-small.txt Customer-small.txt|./mapperjoin.py|sort|./reducerjoin.py

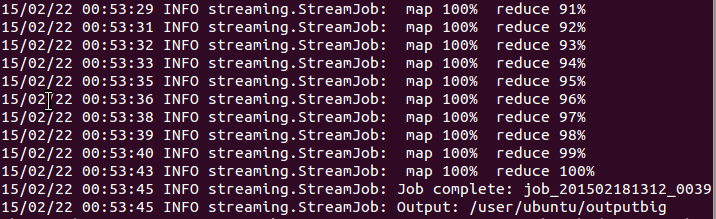


After we make sure it works, then we run the mapper and reducer script using Hadoop with the real data set using following command:

$HADOOP\_HOME/bin/hadoop jar /home/ubuntu/Workspace/hadoop-

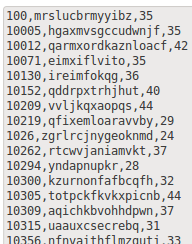
1.1.0/contrib/streaming/hadoop-streaming-1.1.0.jar -file /home/ubuntu/Workspace/examples/python/mapperjoin.py -mapper /home/ubuntu/Workspace/examples/python/mapperjoin.py -file /home/ubuntu/Workspace/examples/python/reducerjoin.py -reducer /home/ubuntu/Workspace/examples/python/reducerjoin.py -input /user/ubuntu/inputbig/\* -output /user/ubuntu/outputbig

Screen capture:



The query resulted **4942 records** (customer which has country code = 5).

Sample output:



Description: Customer ID, Customer Name, Count Transactions

**6. Query 6**

Still, we choose Python to do this query. The mapper function is attached as file ‘PMapper.py’ and the reducer function is ‘PReducer.py’.

To execute the file for the sixth query:

1. Start Hadoop and Pig

2. Clear output folder for query's result: > hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/query6

3. Go to hadoop local directory.

4. Make sure our Mapper and Reducer function is excutable.

5. Specify the number of reducer to be 1 and run: > bin/hadoop jar contrib/streaming/hadoop-streaming-1.1.0.jar -D mapred.reduce.tasks=1 -file /home/ubuntu/Desktop/PMapper.py -mapper /home/ubuntu/Desktop/PMapper.py -file /home/ubuntu/Desktop/PReducer.py -reducer /home/ubuntu/Desktop/PReducer.py -input /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction -output /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/query6

The reason of specifying the reducer number to be 1 is that, the reducers input under the Hadoop streaming mode is just a stream. These means all the Key-Value pairs will fit into all reducers, instead of each Key-Value pair goes to different reducer. So, we limit the number of reducer in order to fit all our result in one file, which is attached as ‘query6result.txt’.

Compared to query1, query6 is more Hadoop visible, more close to MapReduce algorithm, and query1 is more like a SQL language. Also, as described above the input of reducer in query6 is totally different from typical JAVA MapReduce framework.

In a word, the Hadoop steaming mode makes our MapReduce algorithm even difficult to perform the SQL queries. It’s neither more convenient than JAVA nor more efficient than Pig.