CSC 555 Mining Big Data

Assignment 3

Lavinia Wang #1473704

Due Tuesday, May 7th

1) MapReduce:

   a) Describe how to implement the following query in MapReduce

```
SELECT SUM(lo_extendedprice)
FROM lineorder, dwdate
WHERE lo_orderdate = d_datekey
  AND d_yearmonth = 'Jan1994'
  AND lo_discount BETWEEN 4 AND 7;
```

   Mapper1(lineorder): (key:lo_orderdate, value: lo_extendedprice, sourceid)
   For an input block of data from lineorder as the source, for every lineorder record that the code identifies, set lo_orderdate as the key and lo_extendedprice and the source table identifier as the value, but only if the record has a lo_discount value between 4 and 7.

   Mapper2(dwdate): (key:d_datekey, value: sourceid)
   For an input block of data from dwdate as the source, for every dwdate record that the code identifies, set d_datekey as the key and the source table identifier as the value (no other values are needed). The mapper will only output a key-value pair where the record has a d_yearmonth value equaling Jan1994.

   Reducer1: For each lo_orderdate received from Mapper1, pair it with a d_datekey received from Mapper2. If no match is found, discard the Mapper1 key-value. After all keys from Mapper1 are matched or eliminated, sum all of the remaining lo_extendedprice values and output; there will be a single value written. Note, after pairing is complete, the key-values from Mapper2 can be discarded.

   b)

```
SELECT d_month, COUNT(DISTINCT d_sellingseason)
FROM dwdate
GROUP BY d_month
ORDER BY COUNT(DISTINCT d_sellingseason)
```

   Mapper1: (key: d_month, value: d_sellingseason)
   For an input block of data, for every dwdate records that the code identifies, set d_month as the key and set d_sellingseason as a value.
   Reducer1: For each d_month received, output d_month and the count of unique values of d_sellingseason.

   Second pass for sorting, applied to output of previous pass.

   Mapper2: (key: count_d_sellingseason, value: d_month)
   For an input block of data, for each record with the count of distinct d_sellingseason and d_month, set the count of distinct d_sellingseasaon as the key and the corresponding d_month as the value.

Modify the partitioner to a custom range function in order to enable key-based sorting.

Reducer2: For each count_d_sellingseason received, output the d_month values as a list (e.g., 2, June, July). Note that the result is not going to be exactly as the SQL query output because of duplicate count_d_sellingseason entries.

2) Consider a Hadoop job that processes an input data file of size equal to 85 disk blocks (85 different blocks, you can assume that HDFS replication factor is set to 1). The mapper in this job requires 2 minutes to read and fully process a single block of data. Reducer requires 1 second (not minute) to produce an answer for one key worth of values and there are a total of 5000 **distinct** keys (mappers generate a lot of key-value pairs, but keys only occur in the 1-5000 range for a total of 5000 unique entries). Assume that each node has a reducer and that the keys are distributed evenly.

   a) How long will it take to complete the job if you only had one Hadoop worker node? For the sake of simplicity, assume that that only one mapper and only one reducer are created on every node.

      85 * 2 * 60 + 5000 = 15,200 (s)
      4 hours, 13 minutes, 20 seconds

   b) 30 Hadoop worker nodes?

      15200 / 10 = 1520 (s)
      25minutes, 20 seconds

   c) 50 Hadoop worker nodes?

      15200 / 50 = 304 (s)
      5 minutes, 4 seconds

   d) 100 Hadoop worker nodes?

      15200 / 100 = 152 (s)
      2 minutes, 32 seconds

   e) Do you expect the introduction of the combiner affect the runtime of this job? Why or why not? (no credit for a "yes/no" answer, you have to explain).

      Although we are not considering network transfer costs for our calculations here, the Combiner performs aggregation on the output from the mappers on the same nodes as the Mappers. As a result of aggregation, fewer records are transferred over the network to the Reducers, reducing network transfer cost.
      Also, the amount of data to be processed by the Reducers will be reduced.

   f) Would changing the replication factor have any affect your answers for a-d?

The write time of the reduce tasks results will increase.

You can ignore the network transfer costs as well as the possibility of node failure.

3)

a) Suppose you have a 6-node cluster with replication factor of 3. Describe what MapReduce has to do after it determines that a node has crashed while a job is being processed. For simplicity, assume that the failed node is not replaced and your cluster is reduced to 5 nodes. Specifically:

i) What does HDFS (the storage layer) have to do in response to node failure in this case?

The NameNode will direct the replication of the blocks that were on the dead node to the remaining nodes. In this case, replication factor is 3, so there are still two copies of the blocks remaining that were previously on the failed node.

ii) What does MapReduce engine have to do to respond to the node failure? Assume that there was a job in progress because otherwise MapReduce does not need to do anything.

Any failed Map tasks must be restarted. These failed tasks will be set to idle by the Master and will be run on one of the remaining Workers once available. For a failed Reducer, its currently executing Reduce tasks are set to idle and rescheduled to run on another reducer.

b) Where does the Mapper store output key-value pairs before they are sent to Reducers?

On the Mapper's node.

c) Can Reducers begin processing before Mapper phase is complete? **Why or why not?**

The Reducers need the entire dataset before they can begin processing. For example, the Reducers for a word count job cannot count the words until all of the words are present.

4) Using the SSBM schema (http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql) load the Part table into Hive (data available at http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl)

NOTE: The schema above is made for Hive, but by default Hive assumes '\t' separated content. You will need to modify your CREATE TABLE statement to account for '|' delimiter in the data.

Use Hive user defined function (i.e., SELECT TRANSFORM from Lecture4, weekday mapper is available here: http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/weekday_mapper.py) to perform the

following transformation on Part table (creating a new PartSwap table): in the 2nd column/p_name swap the two words in the column and add the length of the shortest word. For example, rose moccasin would become moccasin rose4 or honeydew dim would be dim honeydew3. Your transform python code (split/join) should **always** use tab ('\t') between fields even if the source data is |-separated.

Create Table Commands:

```
create table part (
p_partkey     int,
p_name        varchar(22),
p_mfgr        varchar(6),
p_category    varchar(7),
p_brand1      varchar(9),
p_color       varchar(11),
p_type        varchar(25),
p_size        int,
p_container   varchar(10) )
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

create table part2 (
p_partkey     int,
p_name1       varchar(22),
p_mfgr        varchar(6),
p_category    varchar(7),
p_brand1      varchar(9),
p_color       varchar(11),
p_type        varchar(25),
p_size        int,
p_container   varchar(10) )
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' STORED AS TEXTFILE;
```

assignment3.py

```
#!/usr/bin/python
import sys

for line in sys.stdin:
        line = line.strip().split('\t')
        name = line[1].strip().split(' ')
        length = 0
        if len(name[0]) < len(name[1]):
                length = len(name[0])
        else length = len(name[1])
        print '\t'.join([line[0], name[1], name[0], \
                        line[2], line[3], line[4], line[5], \
                        line[6], line[7], line[8]])
```

5) Download and install Pig:

```
cd
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/pig-0.15.0.tar.gz
gunzip pig-0.15.0.tar.gz
tar xvf pig-0.15.0.tar
```

set the environment variables (this can also be placed in ~/.bashrc to make it permanent)

```
export PIG_HOME=/home/ec2-user/pig-0.15.0
export PATH=$PATH:$PIG_HOME/bin
```

Use the same vehicles file. Copy the vehicles.csv file to the HDFS if it is not already there.

Now run pig (and use the pig home variable we set earlier):

```
cd $PIG_HOME
bin/pig
```

Create the same table as what we used in Hive, assuming that vehicles.csv is in the home directory on HDFS:

```
VehicleData = LOAD '/user/ec2-user/vehicles.csv' USING PigStorage(',')
AS (barrels08:FLOAT, barrelsA08:FLOAT, charge120:FLOAT, charge240:FLOAT, city08:FLOAT);
```

You can see the table description by

```
DESCRIBE VehicleData;
```

Verify that your data has loaded by running:

```
VehicleG = GROUP VehicleData ALL;
Count = FOREACH VehicleG GENERATE COUNT(VehicleData);
DUMP Count;
```

How many rows did you get? (if you get an error here, it is likely because vehicles.csv is not in HDFS)

34,174

Create the same ThreeColExtract file that you have in the previous assignment, by placing barrels08, city08 and charge120 into a new file using PigStorage .You want the STORE command to record output in HDFS. (discussed in p457, Pig Chapter, “Data Processing Operator section)

NOTE: You can use this to get one column:

```
OneCol = FOREACH VehicleData GENERATE barrels08;
```

Verify that the new file has been created and report the size of the newly created file. (you can use **quit** to exit the grunt shell)

File size = 627,873 bytes

```
[ec2-user@ip-172-31-29-137 ~]$ hadoop fs -ls /user/ec2-user/ThreeColExtract
Found 1 items
-rwxr-xr-x   1 ec2-user supergroup     627873 2019-04-30 00:00 /user/ec2-user/ThreeColExtract/000000_0
[ec2-user@ip-172-31-29-137 ~]$
```

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 3" at the top.