Azat Dovgeldiyev

CSC 555

Assignment 3

Problem 1.

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 = 'Feb1995'
  AND lo_discount = 9;
```

First mapper applies to lineorder where key is lo_orderdate and value is lo_extendedprice. For each record of lineorder lo_dicount will be checked if it is 9. Constant lo is to recognize records from lineorder table during the reducer phase.

Mapper 2 applies to dwdate where key is d_datekey. For each record of dwdate it will check if the d_yearmonth is set to Feb1995. If condition meet requirements the mapper will output d_datekey as key and constant value d to identify records from dwdate in the reducer phase.

Reducer 1, for each lo_orderdate received from Mapper 1, paired with a d_datekey received from Mapper 2. Mapper 1 will be discarded if no match is found. Keys from Mapper 1 are matched, sum all the remaining lo_extendedprice values and output, with single written value. After complete pairing, the key-values from Mapper 2 can be discarded.

b.

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

Mapper1: (key: d_month, value: rows of dwdate)
For an input block of data, for every dwdate record that code identifies, set the d_month as key and rows selected rows as value.
Reducer 1: For each d_month received, output d_month and the count of values of rows.

Second pass (required for sorting) -applied to the output of previous pass.

Mapper 2: (key: count(*), value: d_month)

For an input block of data, for each record with count of rows and d_month, set the count of rows as the key and the corresponding d_month as value.

Modify the partitioner to a custom range function to enable key-based sorting.
For each count of rows received, output the d_month values as a list (e.g, March, April).

Problem 2.

a. How long will it take to complete the job if you only had one Hadoop worker node? For simplicity, assume that that only one mapper and only one reducer are created on every node.
The mapper goes every block , so it is 60 *1 = 60 mins, since reducer requires 1 second to produce an answer, in total 7000 seconds, convert it to mins 7000/60 = 116.66 mins, and total **116.66 + 60 = 176.66 mins or 2 hours, 56 mins and 39 secs.**

b. 30 Hadoop worker nodes?
30 worker nodes, 60/30 = 2, without any failure 2*1 = 2 mins. For 30 reducers 7000/30 = 233.33 secs which is 3 mins 53 secs and to sum up, **5 mins and 53 secs.**

c. 50 Hadoop worker nodes?
For 50 worker nodes, 60/50 = 1.2, and with one round 1*2 = 2. 7000/50 = 140 secs or 2 mins 20 secs, in total it will take **4 mins and 10 secs** to complete the job.

d. 100 Hadoop worker nodes?
For 100 worker nodes, 60/100 = 0.6 or 1 min, and with one round 1*1 = 1. 7000/100 = 70 secs or 1 min 10 secs, in total it will take **2 mins and 10 secs** to complete the job.

e. Would changing the replication factor have any affect your answers for a-d?
Since replication factor makes sure whether the node failure keep the data and can assign worker into other nodes, it will **not affect** answers.

Problem 3.

a. **i.** What does HDFS (the storage layer) have to do in response to node failure in this case?
The storage layer will direct the replication of the blocks that were in dead node to the remaining nodes. So master node will replicate into 9 nodes.

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, failed tasks will be set to idle by the Master and will be run on one of the remaining Workers once available, and for a failed Reducer, currently executing Reducer tasks are set to idle and rescheduled to run on another reducer.

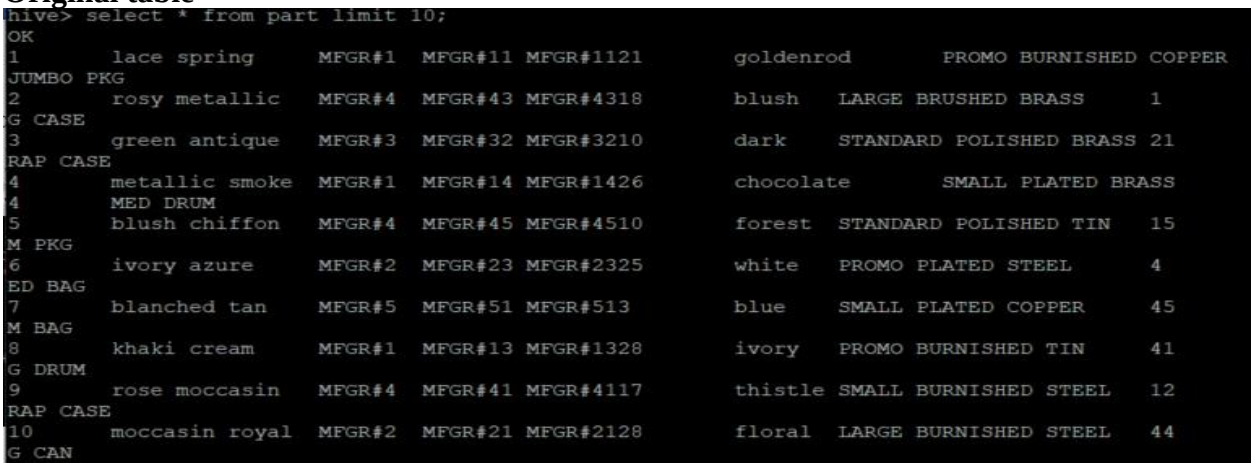
b. Where does the Mapper store output key-value pairs before they are sent to Reducers? Mapper stored data on local disks or called Intermediate files.

c. Can Reducers begin processing before Mapper phase is complete? **Why or why not?** Reducers need entire dataset before they begin processing. Reducers for a word count job cannot count the words until all of the words are present. Reducer will only start when all the mapper job is done.

Problem 4.

Logging initialized using configuration in jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/hive-common-2.0.1.jar!/hive-log4j2.properties
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
SELECT TRANSFORM (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) USING 'python partmapper.py' AS (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) FROM part;

Original table



Transformed table

```
199993  magenta grey    MFGR#3  MFGR#33 MFGR#3326       orange  COPPER~POLISHED~ECONOMY 28
G BAG
199994  aquamarine chiffon      MFGR#1  MFGR#14 MFGR#1414       frosted COPPER~BRUSHED~LARGE
2       MED CAN
199995  dodger magenta  MFGR#4  MFGR#45 MFGR#458        blanched        TIN~POLISHED~PROMO
0       WRAP CAN
199996  steel cyan      MFGR#4  MFGR#44 MFGR#4418       chocolate       COPPER~PLATED~PROMO
1       MED PACK
199997  azure snow      MFGR#4  MFGR#44 MFGR#4426       drab    NICKEL~PLATED~PROMO     37
M DRUM
199998  misty plum      MFGR#5  MFGR#55 MFGR#5512       peach   BRASS~BURNISHED~MEDIUM  49
G BOX
199999  azure cream     MFGR#5  MFGR#52 MFGR#5235       medium  BRASS~PLATED~PROMO      24
G CASE
200000  light midnight  MFGR#5  MFGR#52 MFGR#5223       firebrick       TIN~ANODIZED~MEDIUM
2       LG CAN
Time taken: 21.329 seconds, Fetched: 200000 row(s)
hive>
```

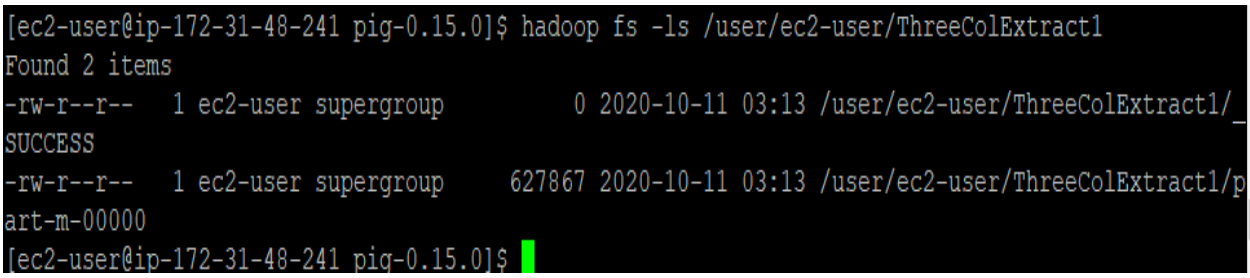
SELECT TRANSFORM (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) USING 'python partmap.py' AS (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) FROM part;

Problem 5.

```
grunt> DESCRIBE VehicleData;
VehicleData: {barrels08: float,barrelsA08: float,charge120: float,charge240: float,city08: float}
grunt> VehicleG = GROUP VehicleData ALL;
grunt> Count = FOREACH VehicleG GENERATE COUNT(VehicleData);
grunt> DUMP Count;
```

```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.4   0.15.0  ec2-user        2020-10-11 01:54:58     2020-10-11 01:55:32     GROUP_BY

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   M
edianReducetime Alias   Feature Outputs
job_1602380992220_0001  1       1       6       6       6       6       4       4       4       4       Count,VehicleData,VehicleG      G
ROUP_BY,COMBINER        hdfs://localhost/tmp/temp325172311/tmp407373796,

Input(s):
Successfully read 34175 records (11766951 bytes) from: "/user/ec2-user/vehicles.csv"

Output(s):
Successfully stored 1 records (9 bytes) in: "hdfs://localhost/tmp/temp325172311/tmp407373796"

Counters:
Total records written : 1
Total bytes written : 9
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

2020-10-11 01:55:32,682 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2020-10-11 01:55:32,682 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(34174)
```

34174 rows.



627867 bytes.

To create file and store it:
OneCol = FOREACH VehicleData GENERATE barrels08, city08, charge120;
Store OneCol INTO ‘ThreeColExtract1’ USING PigStorage(‘,’);