/Alex Teboul - CSC 555 Assignment 2

CSC 555 Mining Big Data

Assignment 2 (due Sunday, 2/2)

Suggested reading: *Mining of Massive Datasets*: Chapter 2.

Hadoop: The Definitive Guide: Chapter 17 (Hive).

1) Describe how you would implement a MapReduce job consisting of Map and Reduce description. You can describe it in your own words or as pseudo-code. Keep in mind that map task reads the input file and produces (key, value) pairs. Reduce task takes a list of (key, value) pairs for each key and combines all values for each key.
Remember that Map operates on individual blocks and Reduce on individual keys with a set of values. Thus, for Mapper you need to state what your code does given a block of data and for Reduce you need to state what your reducer does for each key. You don't have to explain how to parse the file and extract numbers/names.

   a) **For a Student table (ID, Name, Address, Phone, City, State), convert**
      SELECT Year, Month, COUNT(Name)
      FROM Student
      GROUP BY Year, Month;
      *****There isn't even a year or month column in the student table, so it's hard to answer this.
      Map: The mapper creates the pairs of (year + month, name). So the mapper creates/joins year and month, then creates the pairs of year_month key and Name values.
      Reduce: The reducer looks at all the Names for that year_month key and counts the names.

   b) **For Employee(EID, First, Last, Phone, Age) and Agent(AID, First, Last, Address), find everyone with the same name using MapReduce:**
      SELECT a.First, a.Last, EID, AID, Phone
      FROM Employee as e, Agent as a
      WHERE e.Last = a.Last AND e.First = a.First;
      *A bit confused by this query. Are Agents also Employees here? It doesn't seem like they are based on the structure of this query not going on what I assume the primary keys are (EID and AID). Is the query supposed to be displaying employee IDs to match Agent names?
      Map: I assume this would use multiple mappers. Map1 could get the key as first+last with AID values from Agent and Map2 could get the first+last keys and EID, Phone values from Employee. Otherwise maybe you could do the mapping across the tables on those first and last names with the associated EID, AID, Phone and just leave a placeholder on the Employee side.
      Reduce: Then reducer would get combined first+last of both, and values for AID, EID, Phone.

   c) **Same tables:**

SELECT Age, COUNT(DISTINCT a.Last)
FROM Employee, Agent
WHERE EID = AID
GROUP BY Age;
Map: The map1 could get (eid, age, last) and map2 could get (aid, age, last).
Reduce: Then the reduced could see all these ids with age/last and pull only the ones from map2 of Agent where the ids match. Finally it would select the age and count the number of distinct last names that appear for that age by grouping on age. Like is like SQL Join example from class with the student name and faculty area.

2) Suppose you are tasked with analysis of the company's web server logs. The log dump contains a large amount of information with up to 8 different attributes (columns). You regularly run a Hadoop job to perform analysis pertaining to 3 specific attributes – TimeOfAccess, OriginOfAccess and FileName.

a) How would you attempt to speed up the repeated execution of the query? (this is an intentionally open-ended question, there are several acceptable answers)

i) **Answer:** If you're only dealing with those 3 specific attributes and can drop the other 5, you can speed up the query execution. Other common Hive query performance improvement techniques include vectorization, using ORCFile format which compresses and speeds up execution in certain use cases, partitioning the data such that you query partitions, and bucketing with Hive Partition.

b) If a Mapper task fails while processing a block of data – what is the location (which node) where MapReduce framework will prefer to restart it?

i) **Answer:** If the block being processed in a Node fails or is still processing once the other Nodes have finished processing their blocks, then the block that hasn't finished processing will be processed by one of the available Nodes that has a copy of that block. In the in-class example this was Node 2 fails, and Node 1 and 3 process the failed block. The first to finish processing it is then accepted. I think the name node will know where to restart it based on the directory.

c) If the job is executed with 4 Reducers

i) How many files does the output generate?

(1) **Answer:** One output file per reducer, so 4 output files.

ii) Suggest one possible hash function that may be used to assign keys to reducers.

(1) **Answer:** We'd discussed hash functions that involve modulus.

d) True or False?

i) A message that was encrypted with a public key can be decrypted with a corresponding private key

(1) Answer: TRUE

ex. Use someone's public key to encrypt a message that only they can decrypt with their private key.

ii) **A message that was encrypted with a private key can be decrypted with a corresponding public key**
   (1) **Answer: TRUE**
   ex. Encrypt something with your private key and anyone can decrypt with public key to verify authenticity - signature.

iii) **A message that was encrypted with a public key can only be read by its intended recipient, the holder of the private key**
   (1) Answer: TRUE

3) Consider a Hadoop job that processes an input data file of size equal to 45 disk blocks (45 different blocks, you can assume that HDFS replication factor is set to 1). The mapper in this job requires 1 minute to read and fully process a single block of data. For the purposes of this assignment, you can assume that the reduce part of this job takes zero time.

a) Approximately how long will it take to process the file if you only had one Hadoop worker node? You can assume that that only one mapper is created on every node.
   i) **Answer:** 45 minutes = (45block * 1min/1block = 45 min)
   ii) If there's only one mapper on this one worker node, a mapper takes 1minute to process a block, and there are 45 blocks to be read/processed then that's 45 minutes.
   iii) ***I will also assume for all of these that if there's only one mapper on a node, then you can't process fractional components of the blocks either which would speed up processing with more nodes.

b) 20 Hadoop worker nodes?
   i) **Answer:** 3 minutes
   ii) In minute 1, 20 blocks get processed, by minute 2 now 40 have been processed, then by minute 3 the remaining 5 get processed for a total of 3 minutes.

c) 50 Hadoop worker nodes?
   i) **Answer:** 1 minute
   ii) In minute 1, all 45 blocks are being processed by the mapper in their own worker node, so it only takes 1 minute.

d) 75 Hadoop worker nodes?
   i) **Answer:** 1 minute
   ii) There's no advantage to adding more worker nodes at this point because all 45 blocks can be worked on by their own mapper/worker node at 1minute per block = 1minute total. The assumption still being that you can't process fractions of blocks.

e) Now suppose you were told that the replication factor has been changed to 3? That is, each block is stored in triplicate, but file size is still 45 blocks. Which of the answers (if any) in a)-c) above will have to change?
   i) If the rate stays the same at 1 block per minute, then none of the answers would change in this theoretical situation. I don't think it would have to process the blocks multiple times.

You can ignore the network transfer costs and other potential overheads as well as the possibility of node failure. If you feel some information is missing please be sure to state your assumptions.

4) In this section we are going to use Hive to run a few queries over the Hadoop framework. These instructions assume that you are starting from a working Hadoop installation. It should be sufficient to start your instance and the Hadoop framework on it.

   Hive commands are listed in **Calibri bold font**

   a) Download and install Hive:

      cd
      (this command is there to make sure you start from home directory, on the same level as where hadoop is located)
      wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hive-2.0.1-bin.tar.gz
      gunzip apache-hive-2.0.1-bin.tar.gz
      tar xvf apache-hive-2.0.1-bin.tar

      set the environment variables (can be automated by adding these lines in ~/.bashrc). If you don't, you will have to set these variables every time you use Hive.
      export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin
      export PATH=$HIVE_HOME/bin:$PATH

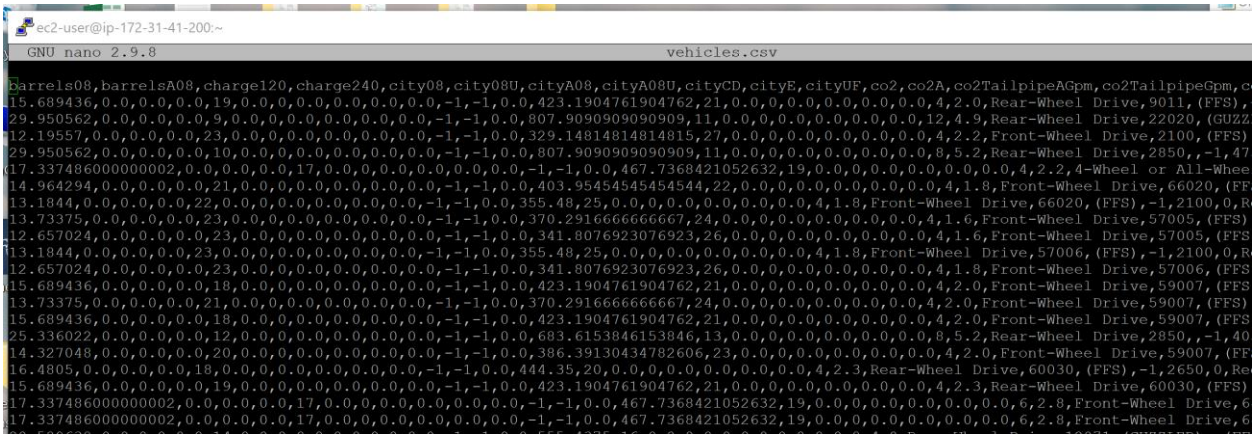
      $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
      $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
      (if you get an error here, it means that /user/hive does not exist yet. Fix that by running **$HADOOP_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse instead)**

      $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
      $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

      We are going to use Vehicle data (originally from http://www.fueleconomy.gov/feg/download.shtml)

      You can get the already unzipped, comma-separated file from here:
      wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/vehicles.csv

      You can take a look at the data file by either
      nano vehicles.csv or
      more vehicles.csv (you can press space to scroll and q or Ctrl-C to break out)



Note that the first row in the data is the list of column names. What follows after commands that start Hive, is the table that you will create in Hive loading the first 5 columns. Hive is not particularly sensitive about invalid or partial data, hence if we only define the first 5 columns, it will simply load the first 5 columns and ignore the rest. You can see the description of all the columns here (atvtype was added later) http://www.fueleconomy.gov/feg/ws/index.shtml#vehicle

Create the ec2-user directory on the HDFS side (absolute path commands should work anywhere and not just in Hadoop directory as bin/hadoop does). Here, we are creating the user "home" directory on the HDFS side.

hadoop fs -mkdir /user/ec2-user/

Run hive (from the hive directory because of the first command below):

cd $HIVE_HOME

$HIVE_HOME/bin/schematool -initSchema -dbType derby

(NOTE: This command initializes the database metastore. If you need to restart/reformat or see errors related to meta store, run **rm -rf metastore_db/** and then repeat the above initSchema command)

bin/hive

You can now create a table by pasting this into the Hive terminal:

```
CREATE TABLE VehicleData (
barrels08 FLOAT, barrelsA08 FLOAT,
charge120 FLOAT, charge240 FLOAT,
city08 FLOAT)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE;
```

You can load the data (from the local file system, not HDFS) using:

```
LOAD DATA LOCAL INPATH '/home/ec2-user/vehicles.csv'
OVERWRITE INTO TABLE VehicleData;
```

```
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.
hive> CREATE TABLE VehicleData (
    > barrels08 FLOAT, barrelsA08 FLOAT,
    > charge120 FLOAT, charge240 FLOAT,
    > city08 FLOAT)
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 1.706 seconds
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/vehicles.csv'
    > OVERWRITE INTO TABLE VehicleData;
Loading data to table default.vehicledata
OK
Time taken: 2.002 seconds
hive>
```

(NOTE: If you downloaded vehicles.csv file into the hive directory, you have to change file name to /home/ec2-user/apache-hive-2.0.1-bin/vehicles.csv instead)

Verify that your table had successfully loaded by running
SELECT COUNT(*) FROM VehicleData;
(Copy the query output and report how many rows you got as an answer.)

```
hive> SELECT COUNT(*) FROM VehicleData;
WARNING: 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.
Query ID = ec2-user_20200203213414_e7b6e238-75cc-48a0-b1a7-35d9df7edcc6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1580764899694_0001, Tracking URL = http://ip-172-31-41-200.ec2.internal:8088/proxy/application_1580764899694_0001/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1580764899694_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-02-03 21:34:29,193 Stage-1 map = 0%,  reduce = 0%
2020-02-03 21:34:36,955 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.3 sec
2020-02-03 21:34:45,786 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.69 sec
MapReduce Total cumulative CPU time: 2 seconds 690 msec
Ended Job = job_1580764899694_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.69 sec   HDFS Read: 11775010 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 690 msec
OK
34175
Time taken: 32.216 seconds, Fetched: 1 row(s)
hive>
```

***** 1 row but a count of 34175 from the query.**
Run a couple of HiveQL queries to verify that everything is working properly:

SELECT MIN(barrels08), AVG(barrels08), MAX(barrels08) FROM VehicleData;
(copy the output from that query)

```
hive> SELECT MIN(barrels08), AVG(barrels08), MAX(barrels08) FROM VehicleData;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usin
g a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20200203215612_8f7df9ce-4f60-4fa8-97e8-c5a0cb207a65
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1580764899694_0003, Tracking URL = http://ip-172-31-41-200.ec2.internal:8088/proxy/appl
ication_1580764899694_0003/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1580764899694_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-02-03 21:56:20,211 Stage-1 map = 0%,  reduce = 0%
2020-02-03 21:56:27,777 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.55 sec
2020-02-03 21:56:35,308 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.9 sec
MapReduce Total cumulative CPU time: 2 seconds 900 msec
Ended Job = job_1580764899694_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.9 sec   HDFS Read: 11777415 HDFS Write: 37 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 900 msec
OK
0.059892        17.820177449476272      47.06831
Time taken: 24.313 seconds, Fetched: 1 row(s)
hive>
```

*****Output shown above**

SELECT (barrels08/city08) FROM VehicleData;

(you do not need to report the output from that query, but report "Time taken")

```
1.1440213918685913
2.4970455169677734
0.9155832926432291
0.550305407980214
0.6539881115867978
0.4898370901743571
0.6278285526093983
0.7875944438733553
0.7163524150848388
0.8716353310479058
0.8716353310479058
1.1440213918685913
Time taken: 0.177 seconds, Fetched: 34175 row(s)
hive>
```

*****Time taken: 0.177 seconds**

Next, we are going to output three of the columns into a separate file (as a way to transform data for further manipulation that you may be interested in)

INSERT OVERWRITE DIRECTORY 'ThreeColExtract'
SELECT barrels08, city08, charge120
FROM VehicleData;

```
hive> INSERT OVERWRITE DIRECTORY 'ThreeColExtract'
    > SELECT barrels08, city08, charge120
    > FROM VehicleData;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different e
xecution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20200203220050_dfa88a5e-a08c-4b5f-b271-91696bb11b2d
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1580764899694_0004, Tracking URL = http://ip-172-31-41-200.ec2.internal:8088/proxy/application_1580764
899694_0004/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1580764899694_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-02-03 22:00:58,485 Stage-1 map = 0%,  reduce = 0%
2020-02-03 22:01:06,120 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.71 sec
MapReduce Total cumulative CPU time: 1 seconds 710 msec
Ended Job = job_1580764899694_0004
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost/user/ec2-user/ThreeColExtract/.hive-staging_hive_2020-02-03_22-00-50_301_742180340876118
2766-1/-ext-10000
Moving data to: ThreeColExtract
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.71 sec   HDFS Read: 11770539 HDFS Write: 627873 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 710 msec
OK
Time taken: 17.988 seconds
hive>
```

You can now exit Hive by running **exit;**

And verify that the new output file has been created (the file will be called 000000_0)
The file would be created <u>in HDFS</u> in user home directory (/user/ec2-user/ThreeColExtract)

```
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -ls /user/ec2-user/ThreeColExtract
Found 1 items
-rwxr-xr-x   1 ec2-user supergroup     627873 2020-02-03 22:01 /user/ec2-user/ThreeColExtract/000000_0
[ec2-user@ip-172-31-41-200 ~]$
```

*****Verified output file has been created**

<u>Report the size of the newly created file and include the screenshot</u>.

```
[ec2-user@ip-172-31-41-200 ~]$ hadoop fs -du /user/ec2-user/ThreeColExtract
627873  /user/ec2-user/ThreeColExtract/000000_0
[ec2-user@ip-172-31-41-200 ~]$
```

***** Roughly 628KB → (627873 bytes)**

Next, you should go back to the Hive terminal, create a new table that is going to load 8 columns instead of 5 in our example (i.e. create and load a new table that defines 8 columns by including columns city08U,cityA08,cityA08U) and use Hive to generate a new output file containing only the city08U and cityA08U columns from the vehicles.csv file. Report the size of that output file as well.

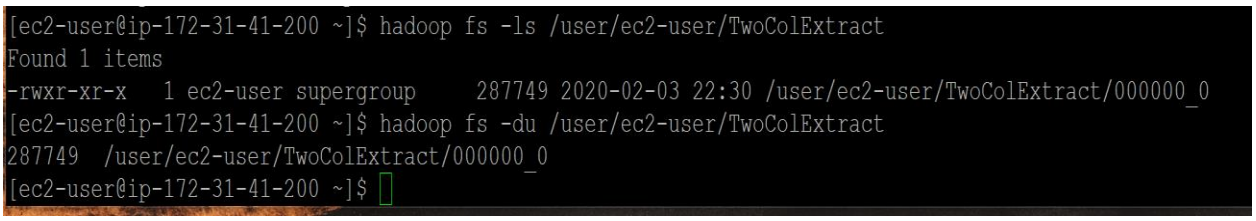
CREATE TABLE VehicleData2 (
barrels08 FLOAT, barrelsA08 FLOAT,

```
charge120 FLOAT, charge240 FLOAT,
city08 FLOAT, city08U FLOAT, cityA08 FLOAT, cityA08U FLOAT)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/vehicles.csv'
OVERWRITE INTO TABLE VehicleData2;

SELECT COUNT(*) FROM VehicleData2;

INSERT OVERWRITE DIRECTORY 'TwoColExtract'
SELECT city08U, cityA08U
FROM VehicleData2;
```



***** File Size shown to be roughly 288KB or 287749 bytes**

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 2" at the top.