**Mining Big Data**

Project, Phase 2

NAME: LI HUANG

# Part 1: Data Transformation

Using A cluster of 4 nodes to process data in this part.(command line in red)

hdfs dfsadmin -safemode leave # leave safemode and remove files with the same name from previous homework

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/lineorder.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/dwdate.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/part.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/supplier.tbl>

Produce a sample first: head -n 100 lineorder.tbl > lineorder.tbl.sample

hadoop fs -put lineorder.tbl /data/lineorder\_s4.tbl

hadoop fs -put lineorder.tbl.sample /data/lineorder\_ss4.tbl

hadoop fs -put dwdate.tbl /data/dwdate\_s4.tbl

hadoop fs -ls /data # checked, everything is fine

**A.**

cd $HIVE\_HOME

bin/home

try on sample dataset first:

CREATE TABLE lineordersl ( lo\_orderkey int, lo\_linenumber int, lo\_custkey int, lo\_partkey int, lo\_suppkey int, lo\_orderdate int, lo\_orderpriority varchar(15), lo\_shippriority varchar(1), lo\_quantity int, lo\_extendedprice int, lo\_ordertotalprice int, lo\_discount int, lo\_revenue int, lo\_supplycost int, lo\_tax int, lo\_commitdate int, lo\_shipmode varchar(10) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl.sample' OVERWRITE INTO TABLE lineordersl;

INSERT OVERWRITE DIRECTORY '/home/ec2-user/lineordersl.csv' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' SELECT \* FROM lineordersl;

DROP TABLE IF EXISTS lineorder;

CREATE TABLE lineorder ( lo\_orderkey int, lo\_linenumber int, lo\_custkey int, lo\_partkey int, lo\_suppkey int, lo\_orderdate int, lo\_orderpriority varchar(15), lo\_shippriority varchar(1), lo\_quantity int, lo\_extendedprice int, lo\_ordertotalprice int, lo\_discount int, lo\_revenue int, lo\_supplycost int, lo\_tax int, lo\_commitdate int, lo\_shipmode varchar(10) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA INPATH '/data/lineorder\_s4.tbl' OVERWRITE INTO TABLE lineorder;

LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl' OVERWRITE INTO TABLE lineorder;

INSERT OVERWRITE DIRECTORY '/user/ec2-user/lineorder.csv' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' SELECT \* FROM lineorder;

Check out result:

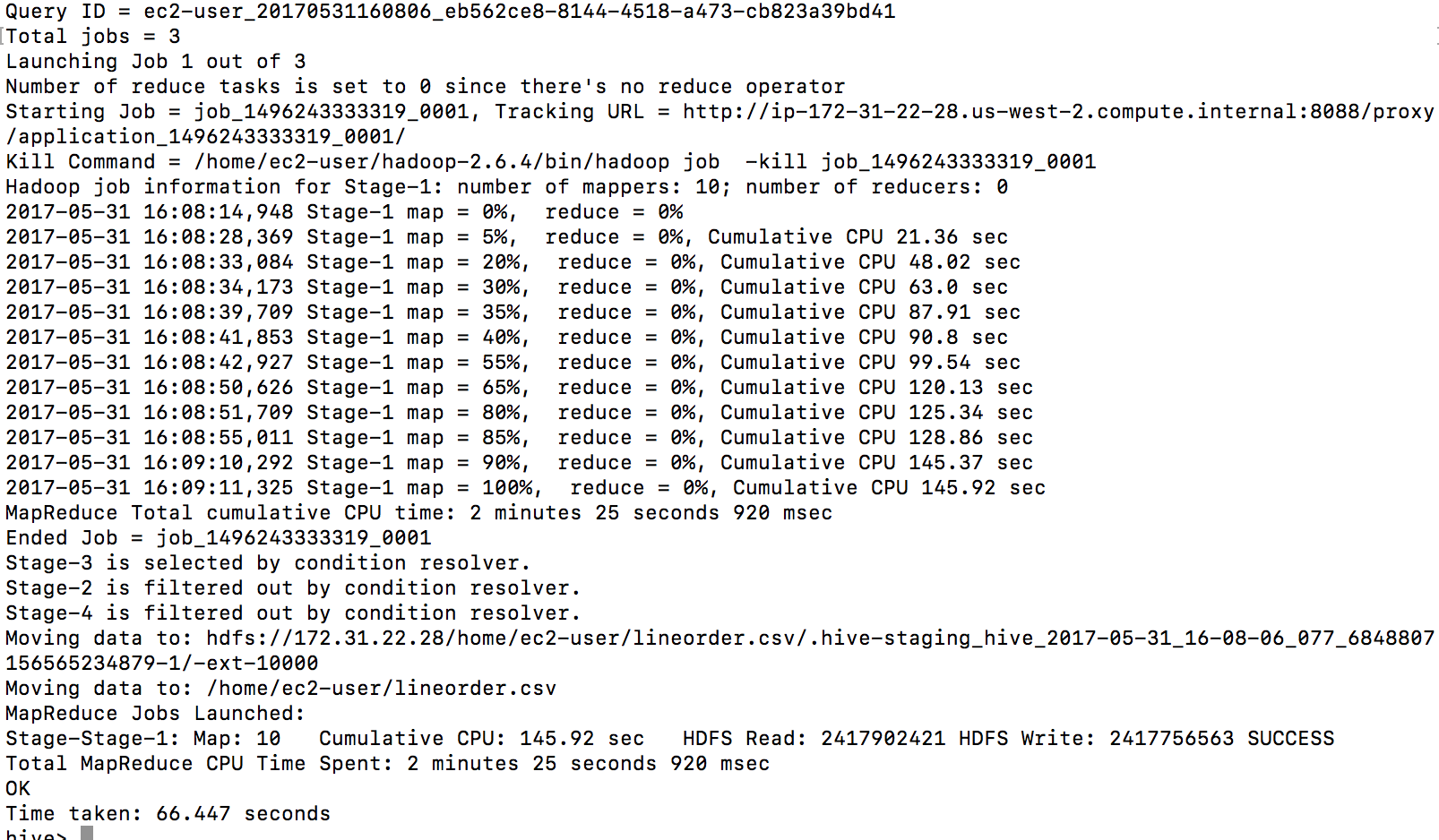
hadoop fs -ls /user/ec2-user/lineorder.csv/

Hadoop fs –head /user/ec2-user/lineorder.csv/

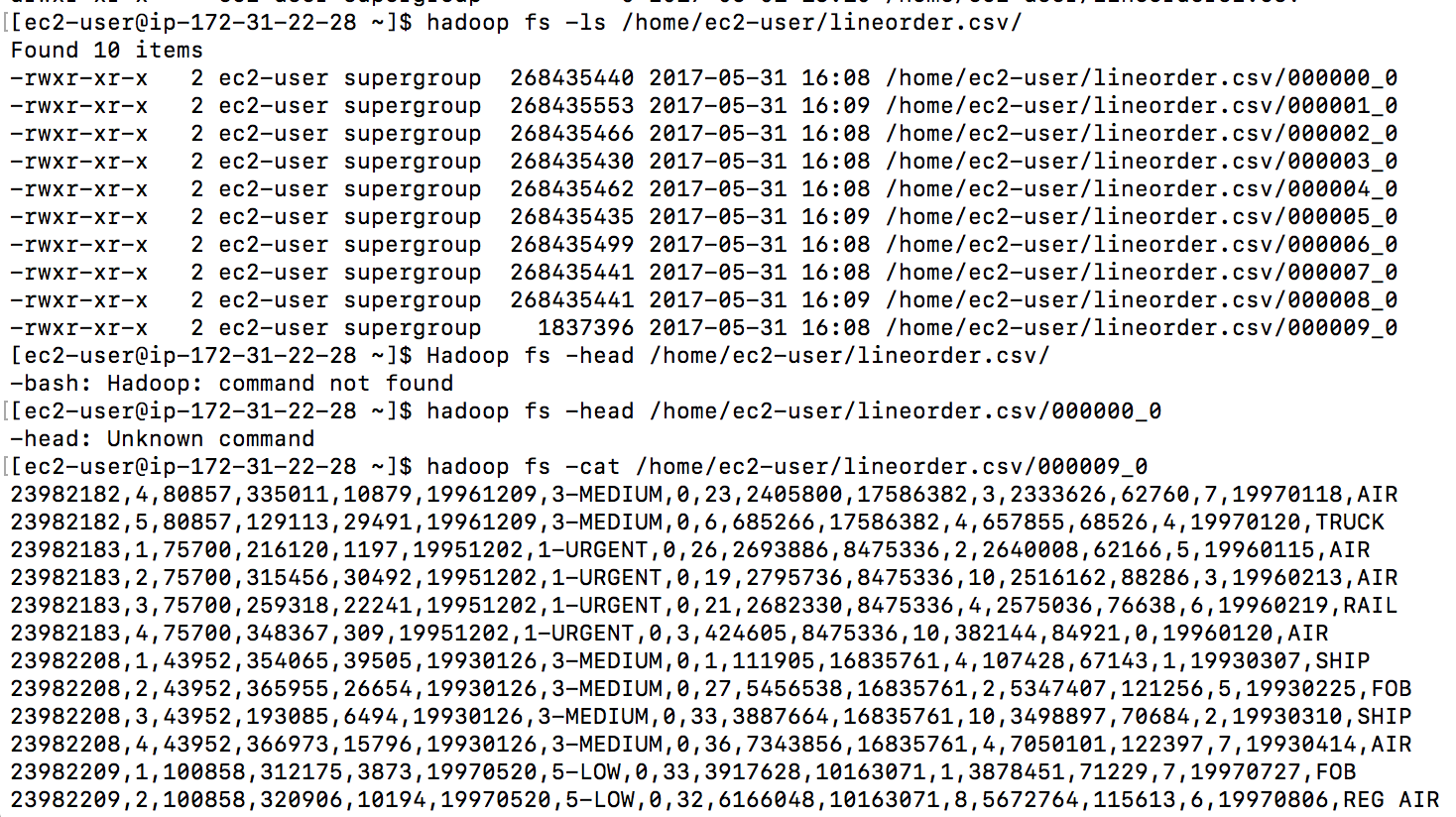
Transform lineorder.tbl table into a csv (comma-separated file)

# TIME ON A SINGLE/CLUSTER NODE FOR QUETSION PART 4. JUST TIME THEM

HIVE:



the csv file:



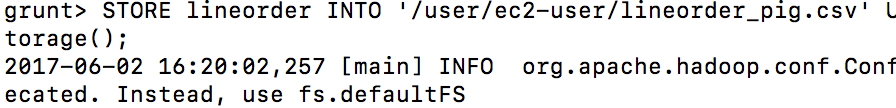
PIG:

lineorder = LOAD '/data/lineorder\_s4.tbl' USING PigStorage('|')AS (lo\_orderkey: int, lo\_linenumber: int, lo\_custkey: int, lo\_partkey: int, lo\_suppkey: int, lo\_orderdate: int, lo\_orderpriority: chararray, lo\_shippriority: chararray, lo\_quantity: int, lo\_extendedprice: int, lo\_ordertotalprice: int, lo\_discount: int, lo\_revenue: int, lo\_supplycost: int, lo\_tax: int, lo\_commitdate: int, lo\_shipmode: chararray);

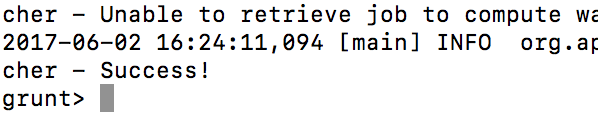
STORE lineorder INTO '/user/ec2-user/lineorder\_pig.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage();

PIG Time Taken:4 min 9 seconds

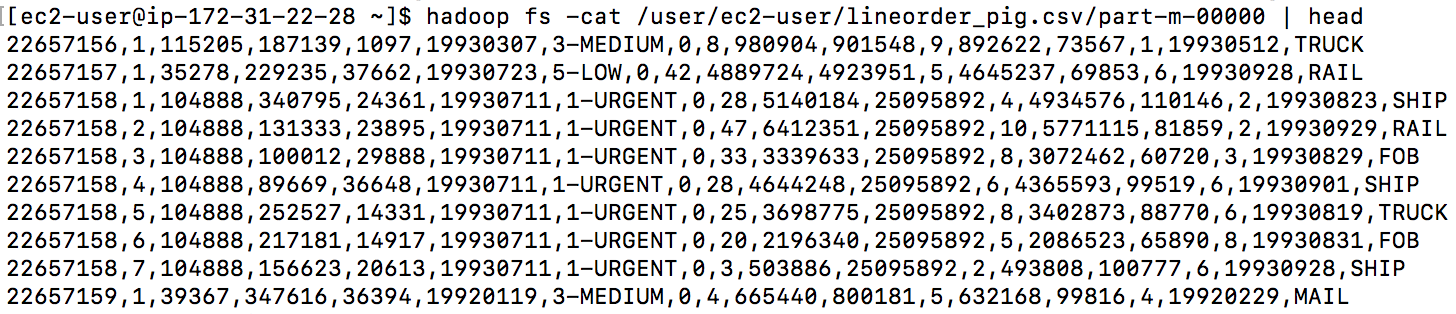
Start



End

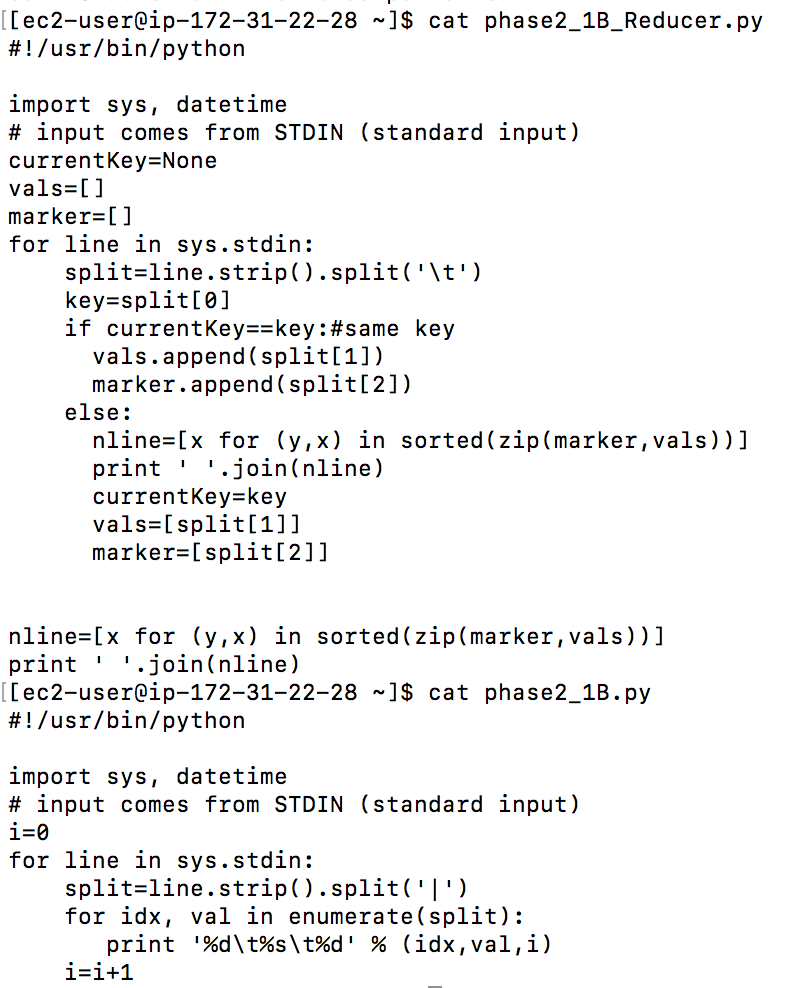


Csv file:



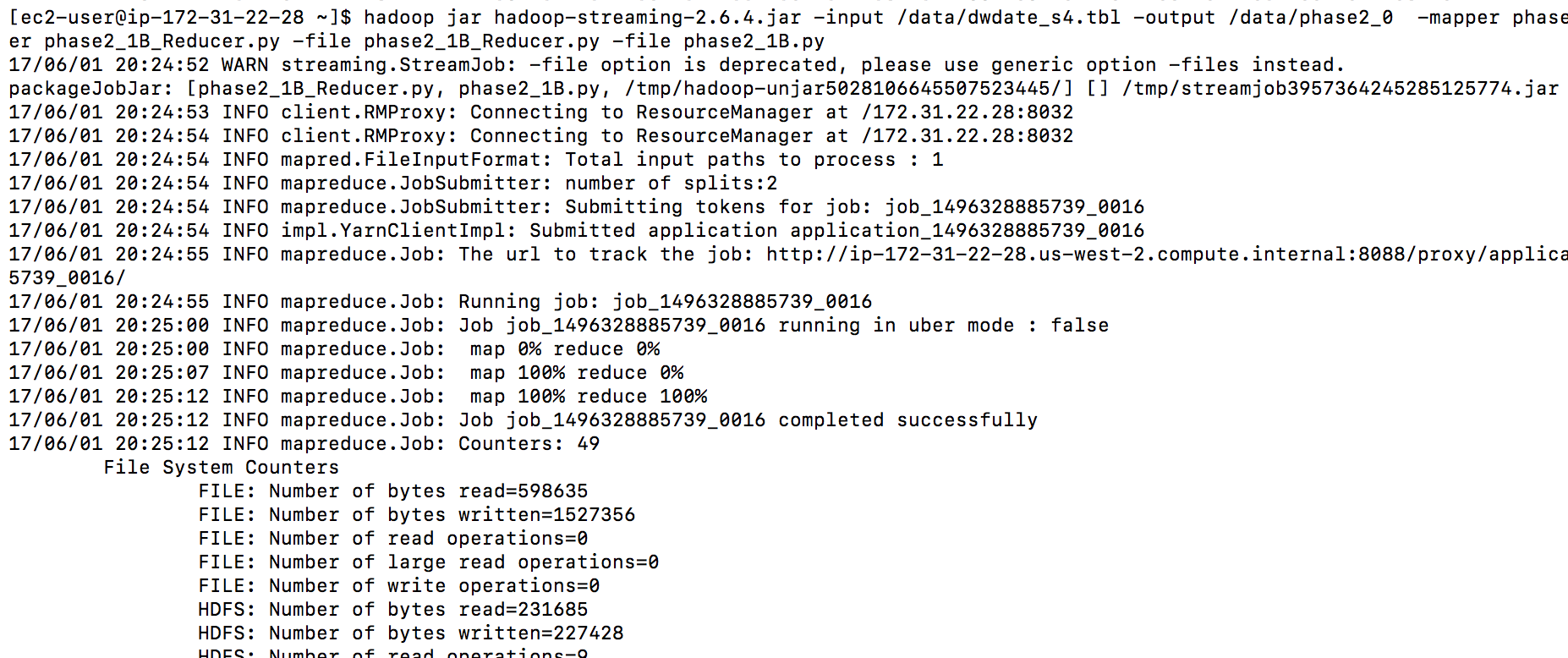
1. Transpose dwdate.tbl table

Python code:



Hadoop jar code:

hadoop jar hadoop-streaming-2.6.4.jar -input /data/dwdate\_s4.tbl -output /data/phase2\_0 -mapper phase2\_1B.py -reducer phase2\_1B\_Reducer.py -file phase2\_1B\_Reducer.py -file phase2\_1B.py



streaming result : ( each column become a line, we can see the first line from the screenshot below)

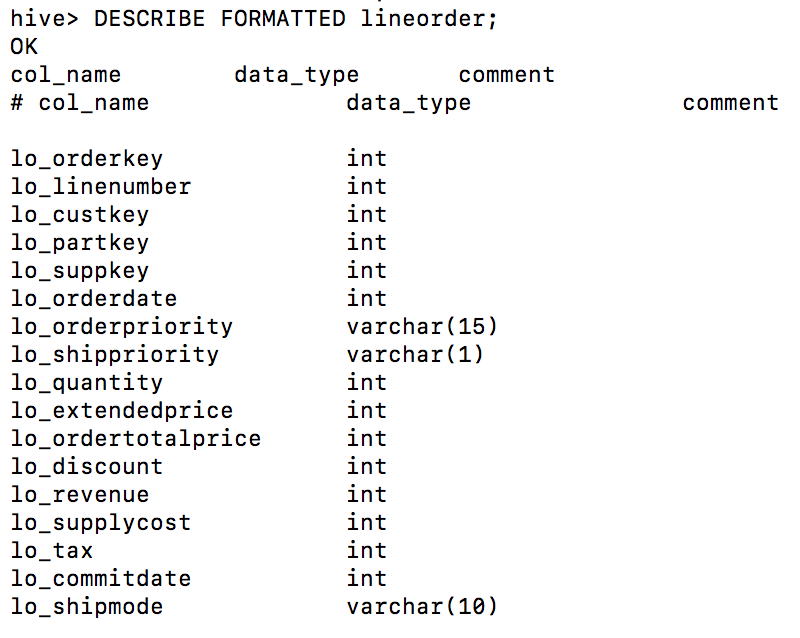


1. Extract all of the numeric columns that **only** contain integer values of < 5000

Return data type for all columns:

DESCRIBE FORMATTED lineorder;

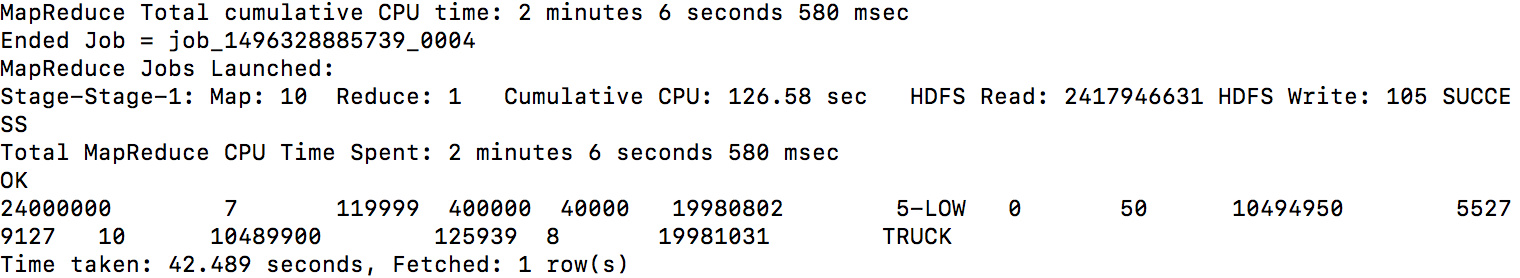
set hive.cli.print.header=true;



Return max value for all columns of lineorder table, if the max value is less than 5000, then extract the column.

Select max(lo\_orderkey), max(lo\_linenumber), max(lo\_custkey), max(lo\_partkey), max(lo\_suppkey), max(lo\_orderdate), max(lo\_orderpriority), max(lo\_shippriority), max(lo\_quantity), max(lo\_extendedprice), max(lo\_ordertotalprice), max(lo\_discount), max(lo\_revenue), max(lo\_supplycost), max(lo\_tax), max(lo\_commitdate), max(lo\_shipmode) from lineorder;

Result:

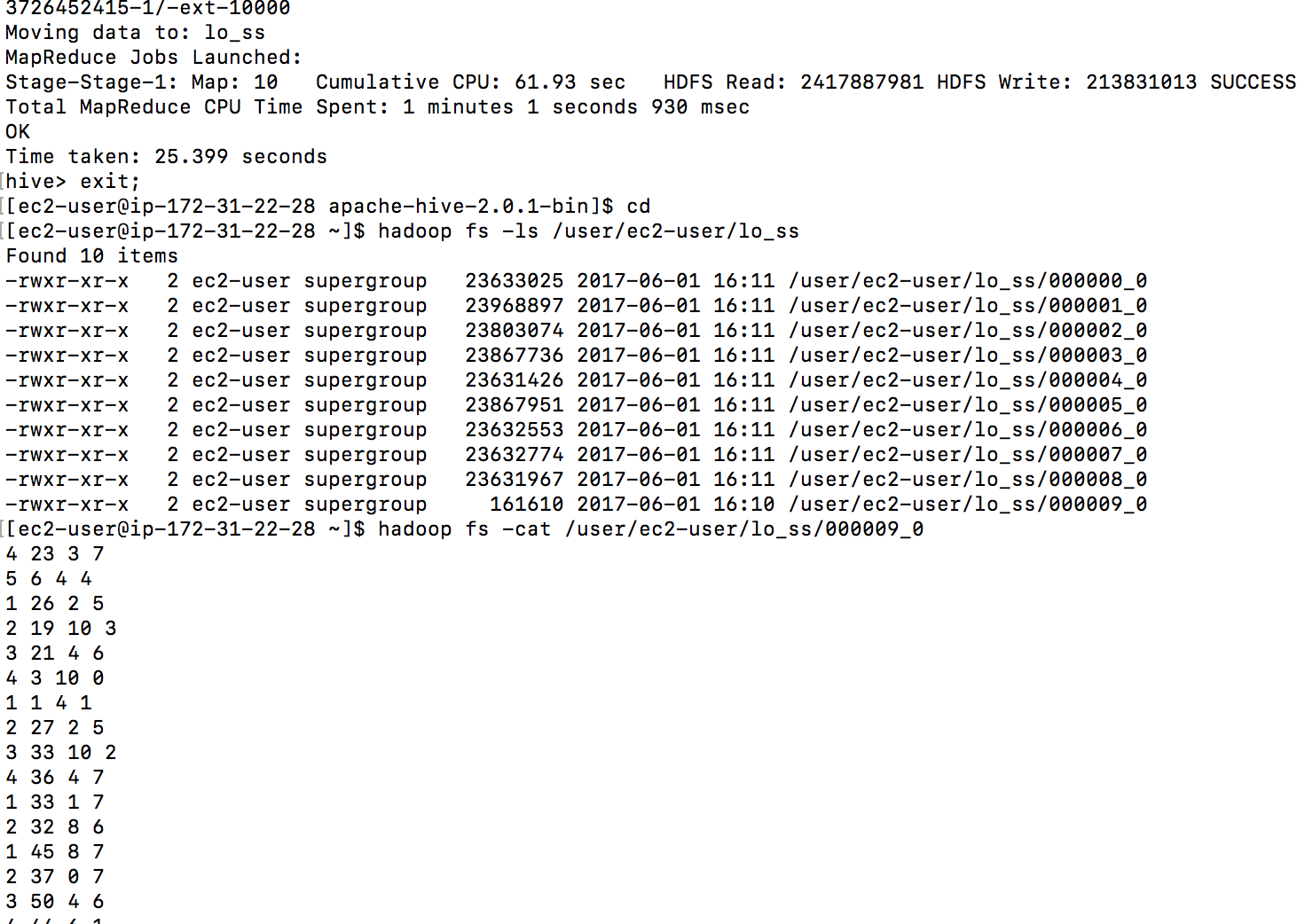


col 2,8,9,13,16, and 8 is not int, so the columns should be extracted are: col(2,5,9,13,16)

lo\_linenumber, lo\_quantity, lo\_discount, lo\_tax

INSERT OVERWRITE DIRECTORY 'lo\_ss' ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ' SELECT lo\_linenumber, lo\_quantity, lo\_discount, lo\_tax FROM lineorder;

Result:

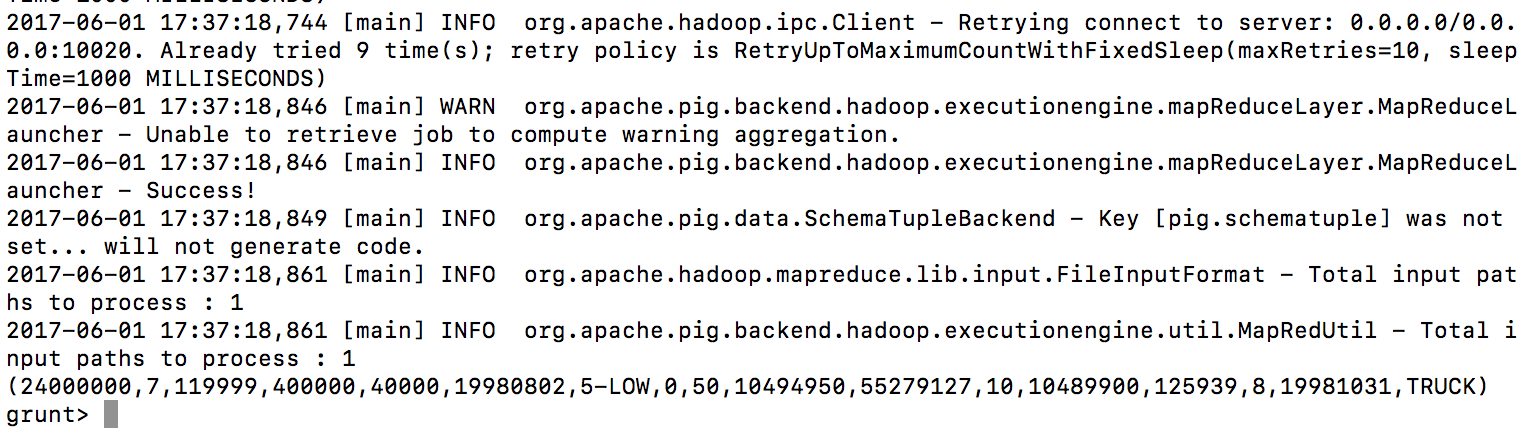


PIG:

LO = GROUP lineorder ALL;

LOMAX = FOREACH LO GENERATE MAX(lineorder.lo\_orderkey), MAX(lineorder.lo\_linenumber), MAX(lineorder.lo\_custkey), MAX(lineorder.lo\_partkey), MAX(lineorder.lo\_suppkey), MAX(lineorder.lo\_orderdate), MAX(lineorder.lo\_orderpriority), MAX(lineorder.lo\_shippriority), MAX(lineorder.lo\_quantity), MAX(lineorder.lo\_extendedprice), MAX(lineorder.lo\_ordertotalprice), MAX(lineorder.lo\_discount), MAX(lineorder.lo\_revenue), MAX(lineorder.lo\_supplycost), MAX(lineorder.lo\_tax), MAX(lineorder.lo\_commitdate), MAX(lineorder.lo\_shipmode);

DUMP LOMAX;



LO\_SS = FOREACH lineorder GENERATE lo\_linenumber, lo\_quantity, lo\_discount, lo\_tax;

STORE LO\_SS INTO '/user/ec2-user/pig\_lo\_ss.csv' USING PigStorage(' ')

Result:



1. Create a pre-join (i.e. a new data file) that corresponds to the following query below.

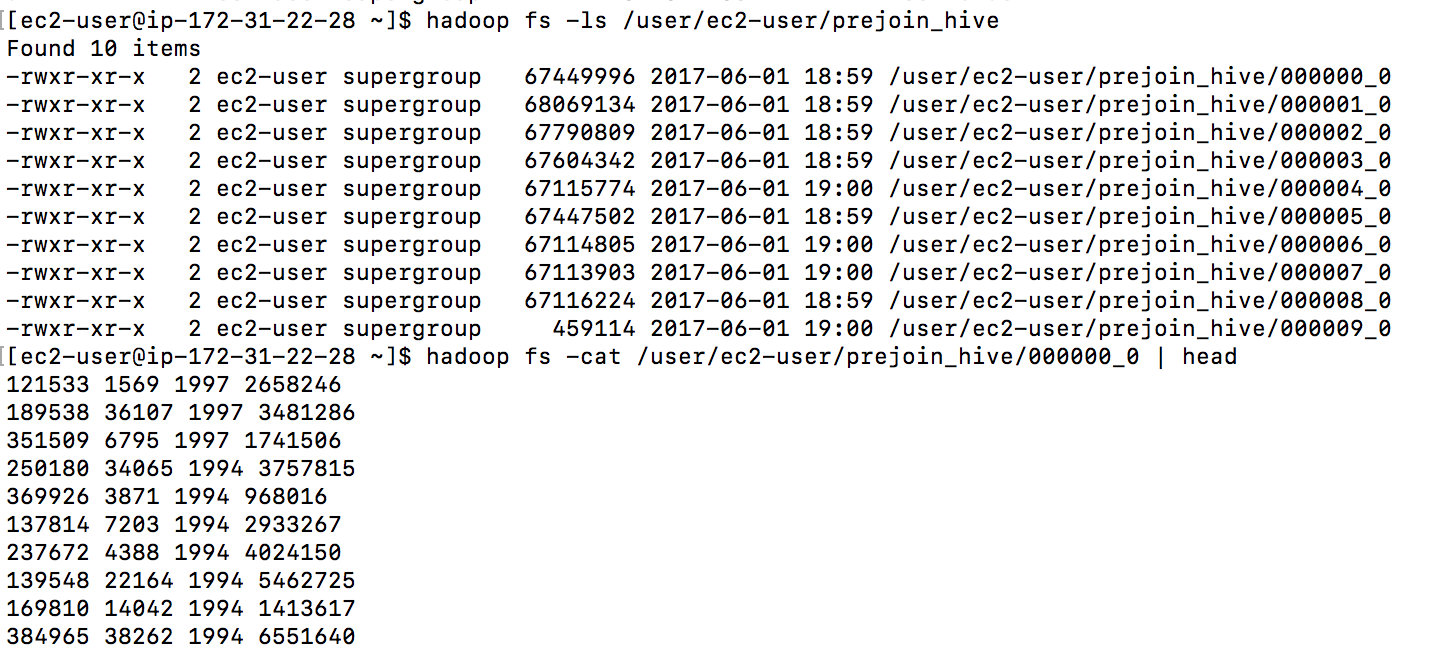
Hive

CREATE TABLE dwdate ( d\_datekey int, d\_date varchar(19), d\_dayofweek varchar(10), d\_month varchar(10), d\_year int, d\_yearmonthnum int, d\_yearmonth varchar(8), d\_daynuminweek int, d\_daynuminmonth int, d\_daynuminyear int, d\_monthnuminyear int, d\_weeknuminyear int, d\_sellingseason varchar(13), d\_lastdayinweekfl varchar(1), d\_lastdayinmonthfl varchar(1), d\_holidayfl varchar(1), d\_weekdayfl varchar(1) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

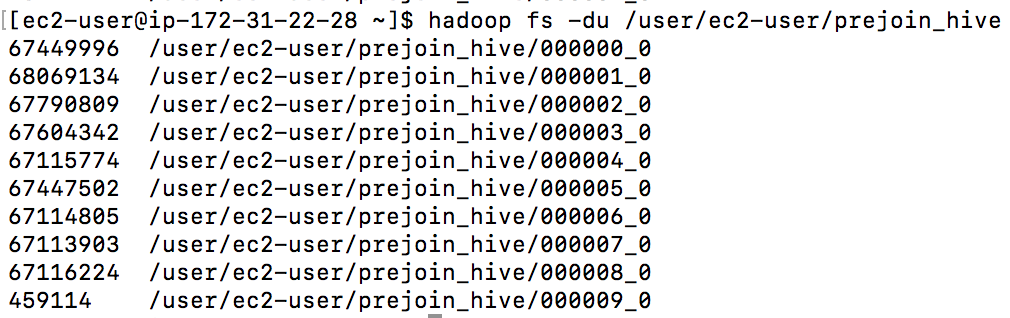
LOAD DATA LOCAL INPATH '/home/ec2-user/dwdate.tbl' OVERWRITE INTO TABLE dwdate;

INSERT OVERWRITE DIRECTORY 'prejoin\_hive' ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ' SELECT lo\_partkey, lo\_suppkey, d\_year, lo\_revenue FROM lineorder, dwdate WHERE lo\_orderdate=d\_datekey;

Prejoin file:



File size:



hadoop fs -du /user/ec2-user/prejoin\_hive/

Pig

We have already created table lineorder; now Create table dwdate:

dwdate = LOAD '/data/dwdate\_s4.tbl' USING PigStorage('|') AS (d\_datekey: int, d\_date: chararray, d\_dayofweek: chararray, d\_month: chararray, d\_year: int, d\_yearmonthnum: int, d\_yearmonth: chararray, d\_daynuminweek: int, d\_daynuminmonth: int, d\_daynuminyear: int, d\_monthnuminyear: int, d\_weeknuminyear: int, d\_sellingseason: chararray, d\_lastdayinweekfl: chararray, d\_lastdayinmonthfl: chararray, d\_holidayfl: chararray, d\_weekdayfl: chararray);

join1 = JOIN lineorder BY lo\_orderdate, dwdate BY d\_datekey;

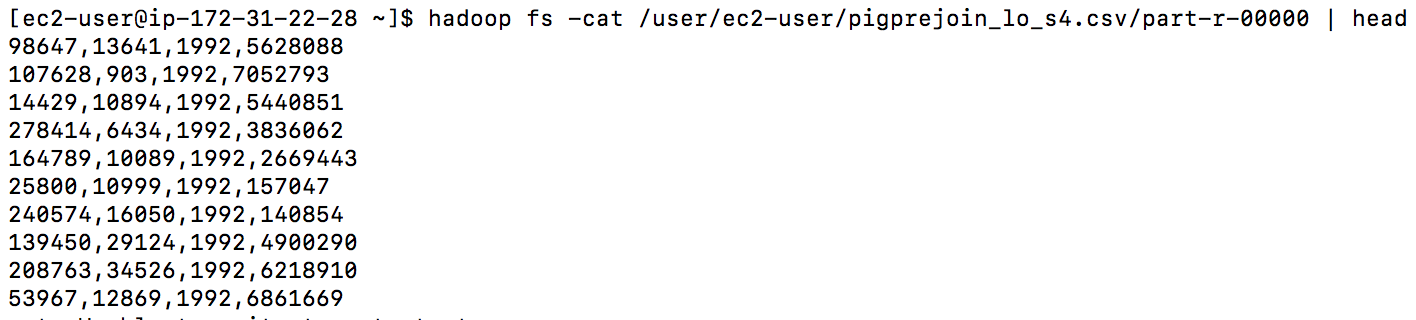
join2 = FOREACH join1 GENERATE lo\_partkey, lo\_suppkey, d\_year, lo\_revenue;

STORE join2 INTO '/user/ec2-user/pigprejoin\_lo\_s4.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage();

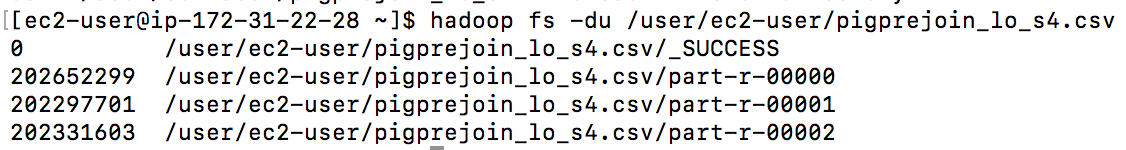
Result:

hadoop fs -ls /user/ec2-user/pigprejoin\_lo\_s4.csv

hadoop fs -cat /user/ec2-user/pigprejoin\_lo\_s4.csv/part-r-00000 | head



file size:



# Part 2: Querying

1. Run SSBM queries 3.1 and 4.1 using any one tool (Hive is the easiest by far)

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/customer.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/supplier.tbl>

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/part.tbl

hadoop fs -put customer.tbl /data/customer\_s4.tbl

hadoop fs -put supplier.tbl /data/supplier\_s4.tbl

hadoop fs -put part.tbl /data/part\_s4.tbl

query 3.1

CREATE TABLE customer ( c\_custkey int, c\_name varchar(25), c\_address varchar(25), c\_city varchar(10), c\_nation varchar(15), c\_region varchar(12), c\_phone varchar(15), c\_mktsegment varchar(10) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;

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;

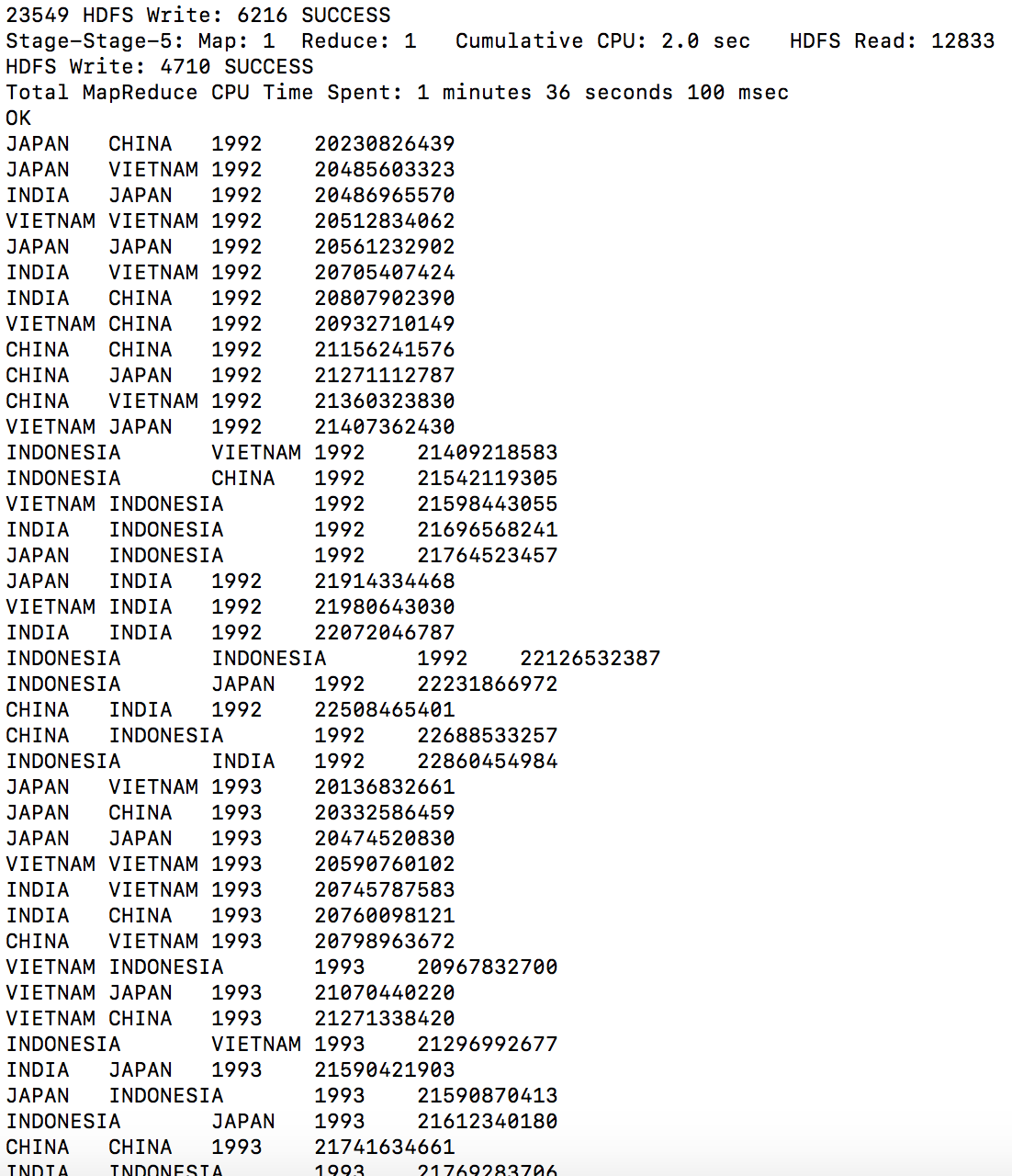
LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl' OVERWRITE INTO TABLE part;

CREATE TABLE supplier ( s\_suppkey int, s\_name varchar(25), s\_address varchar(25), s\_city varchar(10), s\_nation varchar(15), s\_region varchar(12), s\_phone varchar(15) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl' OVERWRITE INTO TABLE supplier;

SELECT c\_nation, s\_nation, d\_year, sum(lo\_revenue) as revenue FROM customer, lineorder, supplier, dwdate WHERE lo\_custkey = c\_custkey and lo\_suppkey = s\_suppkey and lo\_orderdate = d\_datekey and c\_region = 'ASIA' and s\_region = 'ASIA' and d\_year between 1992 and 1997 and d\_year >= 1992 and d\_year <= 1997 GROUP BY c\_nation, s\_nation, d\_year ORDER BY d\_year asc, revenue asc;

Result:



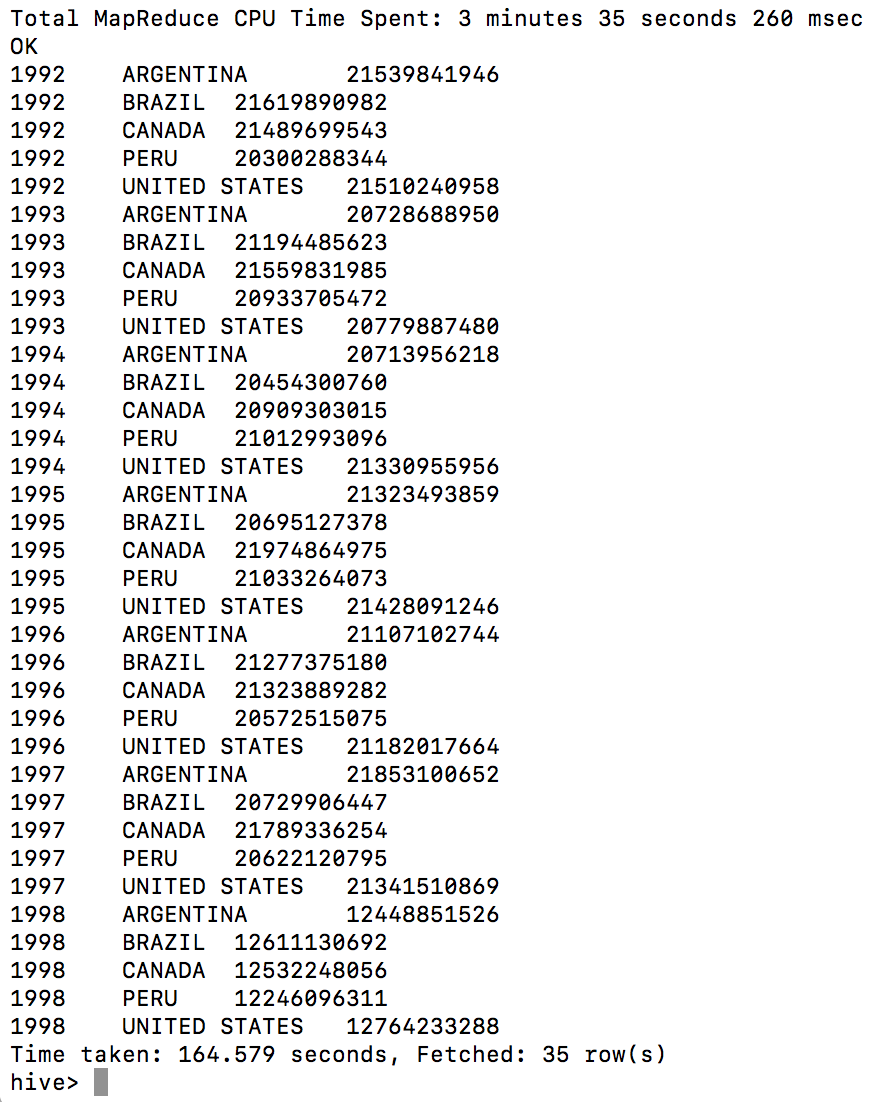
Time taken:



query 4.1

select d\_year, c\_nation, sum(lo\_revenue) as profit1 from lineorder, dwdate, customer, supplier, part where lo\_custkey = c\_custkey and lo\_suppkey = s\_suppkey and lo\_partkey = p\_partkey and lo\_orderdate = d\_datekey and c\_region = 'AMERICA' and s\_region = 'AMERICA' and p\_mfgr = 'MFGR#1' group by d\_year, c\_nation;

result:



1. For this part you should use two of the three choices (Hive, Pig, Hadoop Streaming) to run Q2.1 using what you have created in 1-D

HIVE:

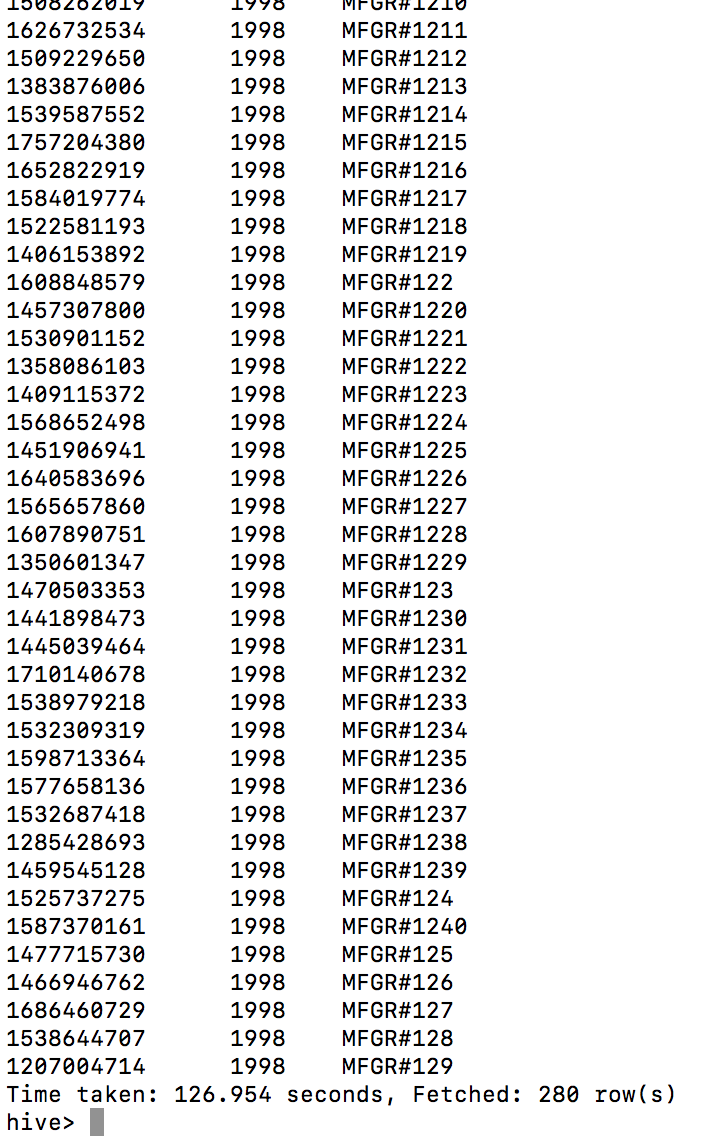
CREATE TABLE ldjoin (lo\_partkey int, lo\_suppkey int, d\_year int, lo\_revenue int)ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ' STORED AS TEXTFILE;

LOAD DATA INPATH '/user/ec2-user/prejoin\_hive' OVERWRITE INTO TABLE ldjoin;

SELECT COUNT(lo\_revenue) FROM ldjoin;

select sum(lo\_revenue), d\_year, p\_brand1 from ldjoin, part, supplier where lo\_partkey = p\_partkey and lo\_suppkey = s\_suppkey and p\_category = 'MFGR#12' and s\_region = 'AMERICA' group by d\_year, p\_brand1 order by d\_year, p\_brand1;

result:



PIG:

hadoop fs -put part.tbl /data/part\_s4.tbl

hadoop fs -put supplier.tbl /data/supplier\_s4.tbl

supplier = LOAD '/data/supplier\_s4.tbl' USING PigStorage('|') AS (s\_suppkey:int, s\_name:CHARARRAY, s\_address:CHARARRAY, s\_city:CHARARRAY, s\_nation:CHARARRAY, s\_region:CHARARRAY, s\_phone:CHARARRAY);

part = LOAD '/data/part\_s4.tbl' USING PigStorage('|') AS (p\_partkey:int, p\_name:CHARARRAY, p\_mfgr:CHARARRAY, p\_category:CHARARRAY, p\_brand1:CHARARRAY, p\_color:CHARARRAY, p\_type:CHARARRAY, p\_size:int, p\_container:CHARARRAY);

ldjoin = LOAD ' /user/ec2-user/pigprejoin\_lo\_s4.csv' USING PigStorage(',') AS (lo\_partkey:int, lo\_suppkey:int, d\_year:int, lo\_revenue:int);

F\_part = FILTER part BY p\_category == 'MFGR#12';

F\_supplier = FILTER supplier BY s\_region == 'AMERICA';

join1= JOIN ldjoin BY lo\_partkey, F\_part BY p\_partkey;

join2= JOIN join1 BY lo\_suppkey, F\_supplier BY s\_suppkey;

GB = GROUP join2 BY (d\_year, p\_brand1);

GE = FOREACH GB GENERATE SUM(join2.lo\_revenue) as revenue, join2.d\_year as d\_year, join2.p\_brand1 as p\_brand1;

RLT = ORDER GE BY d\_year, p\_brand1;

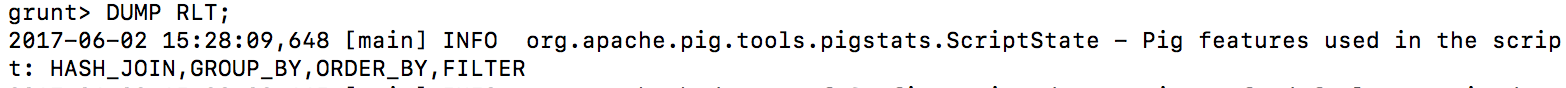
DUMP RLT;

Result:

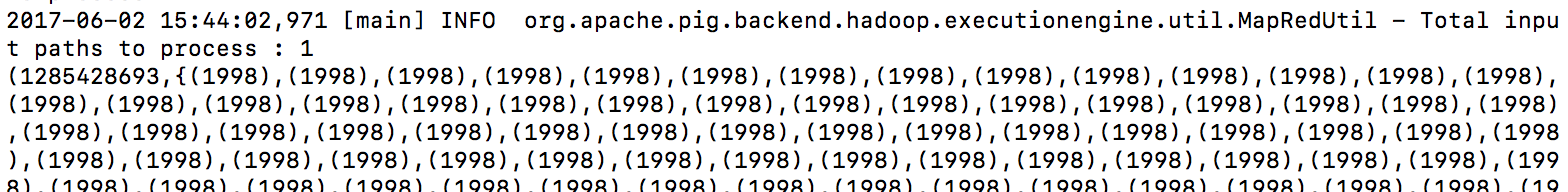


time: 15 min 53 seconds

start



end



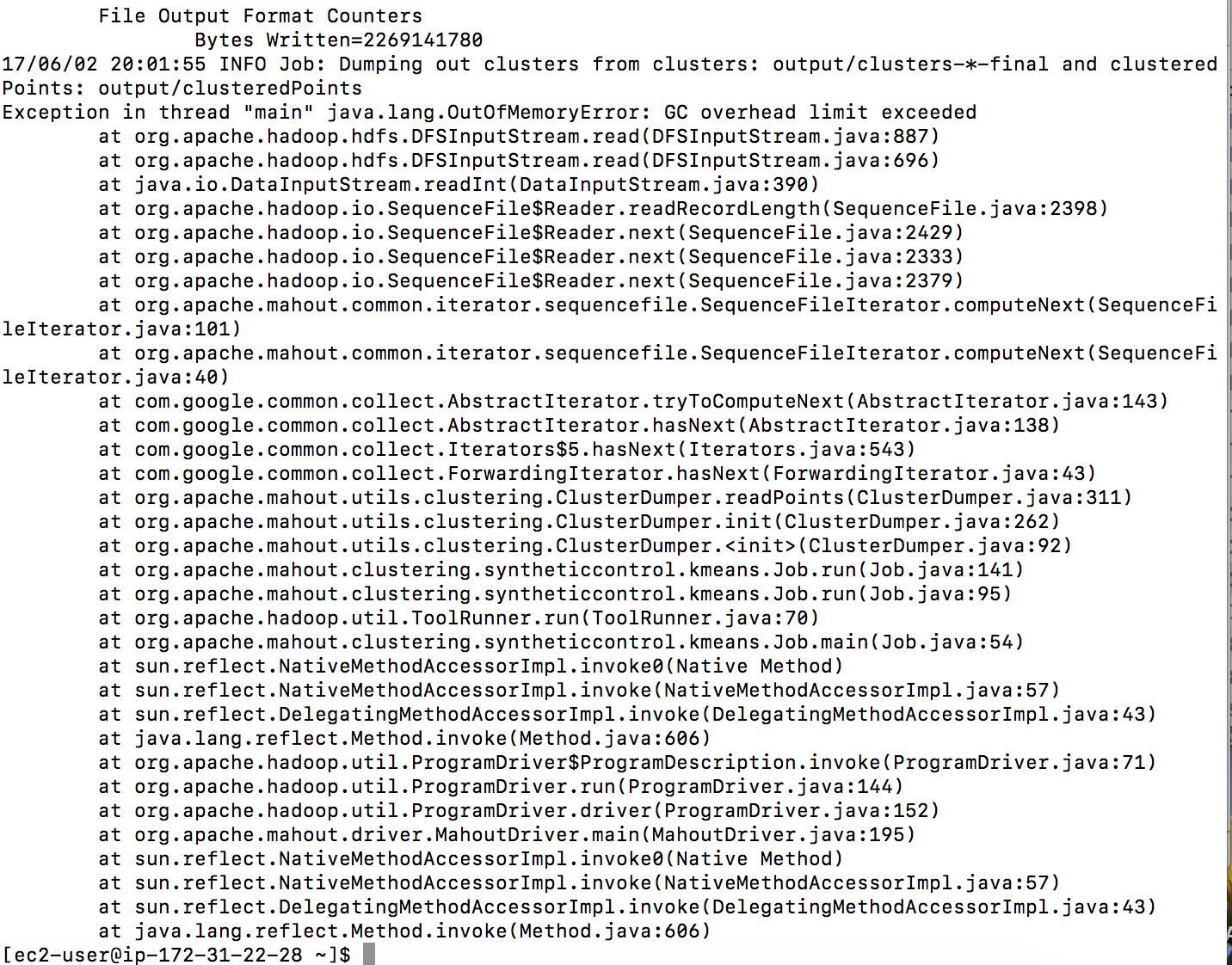
# Part 3: Clustering

Using the file you have created in 1-C, run KMeans clustering using 8 clusters.

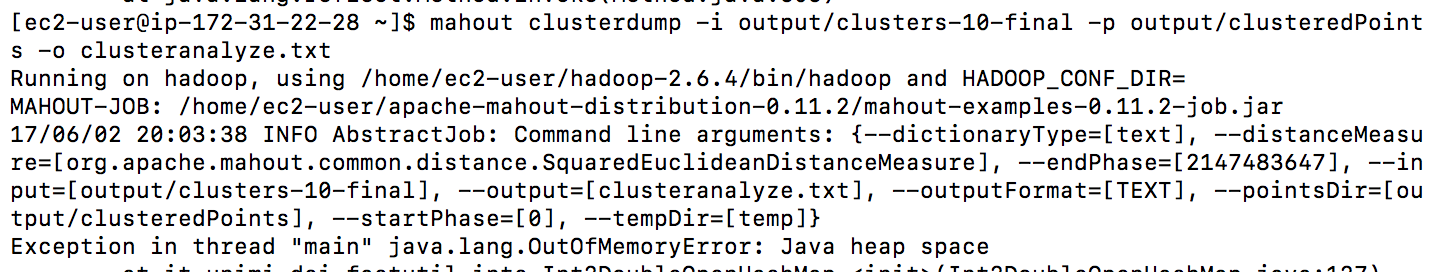
1. Using Mahout as you have in a previous assignment

Copy the data from 1-C hive result to be in the folder testdata so that mahout could read it. And then run the kmeans command as below(k=8):

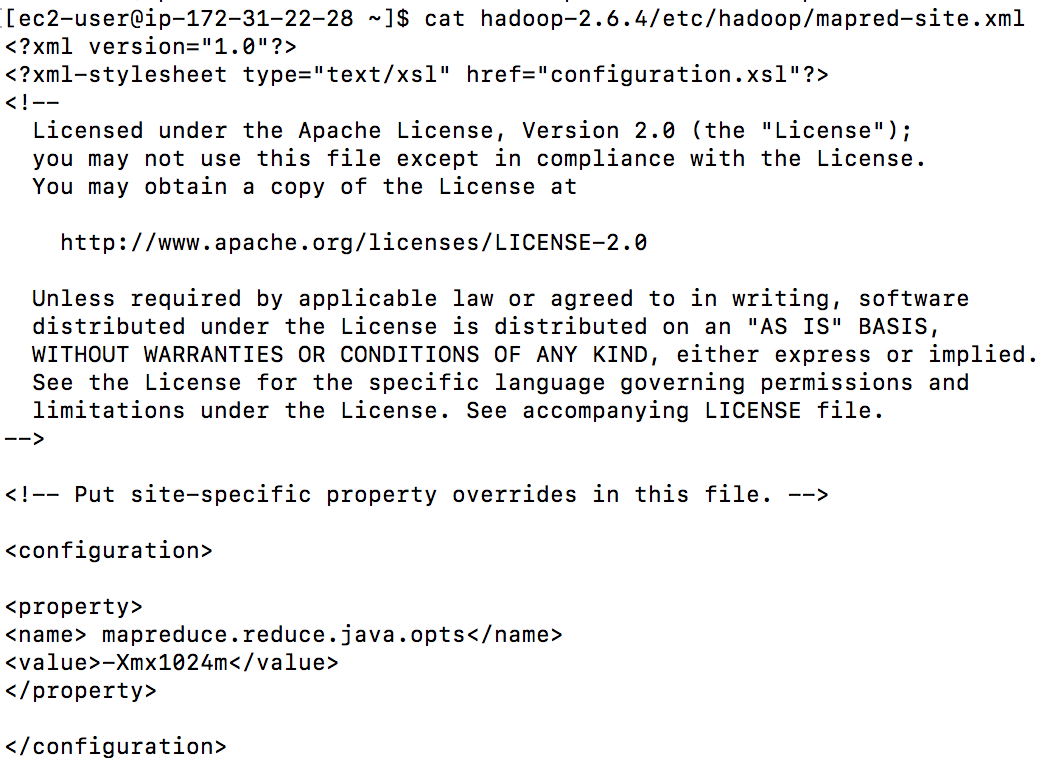
mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job -i testdata -o output -t1 3 -t2 1.5 -cd 0.5 -x 10 -k 8



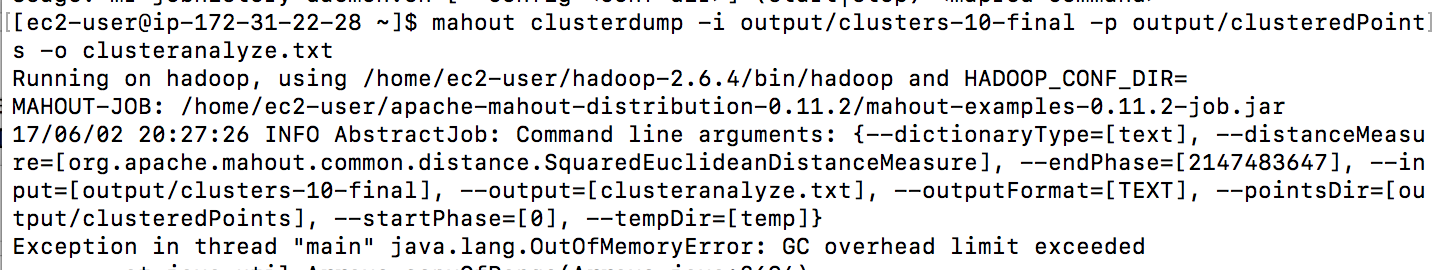
mahout clusterdump -i output/clusters-10-final -p output/clusteredPoints -o clusteranalyze.txt



edit the marped site.xml:



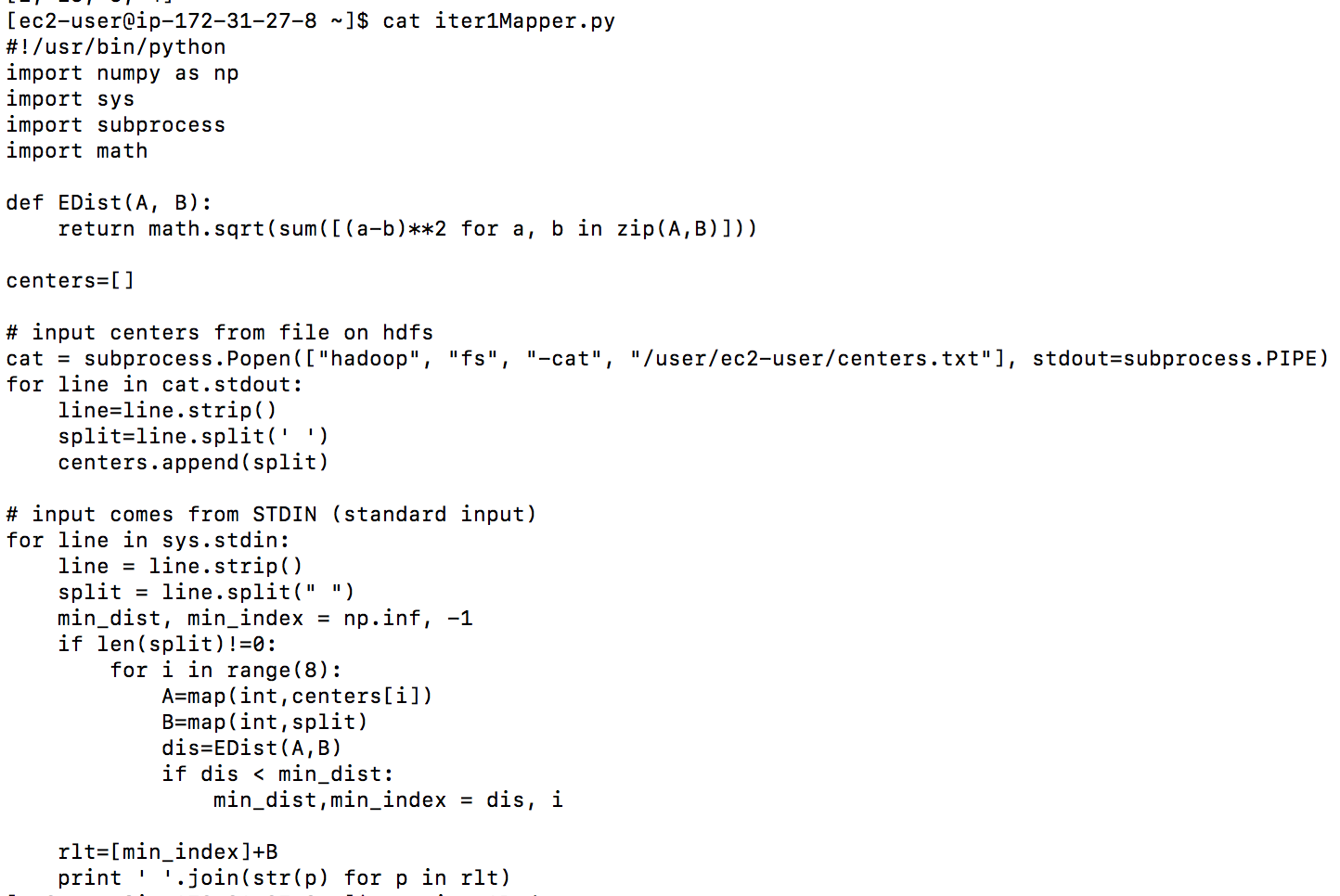
result: still have the same error

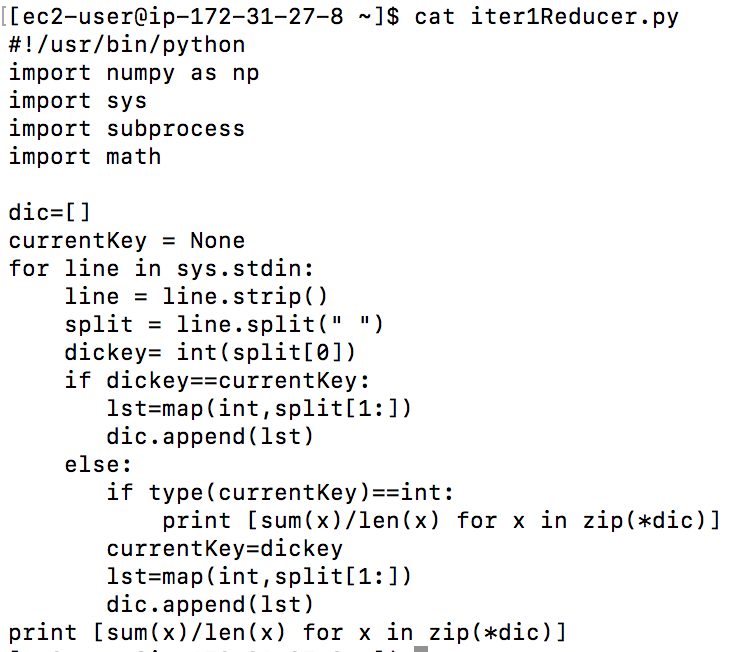


1. Using Hadoop streaming perform two iterations (manually) with randomly chosen input centers.

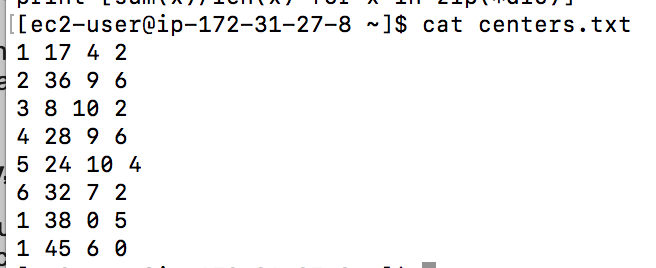
Iteration 1:

Mapper and reducer code:





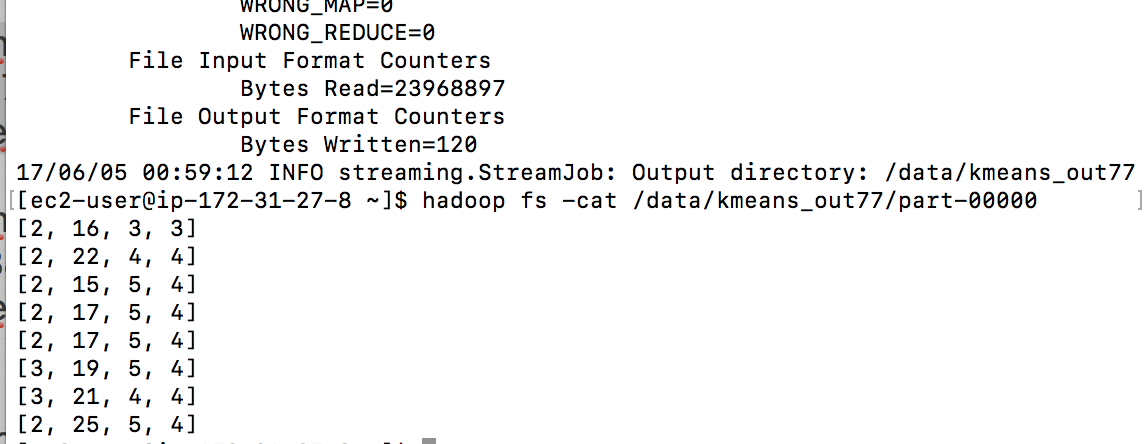
**Center points(8): assigned in file centers.txt**



Test the MR on one file in the input folder first:

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lo\_ss/000000\_0 -output /data/kmeans\_out77 -mapper iter1Mapper.py -reducer iter1Reducer.py -file iter1Reducer.py -file iter1Mapper.py

result: give 8 new centers after iteration 1 for next step’s distance calculation.



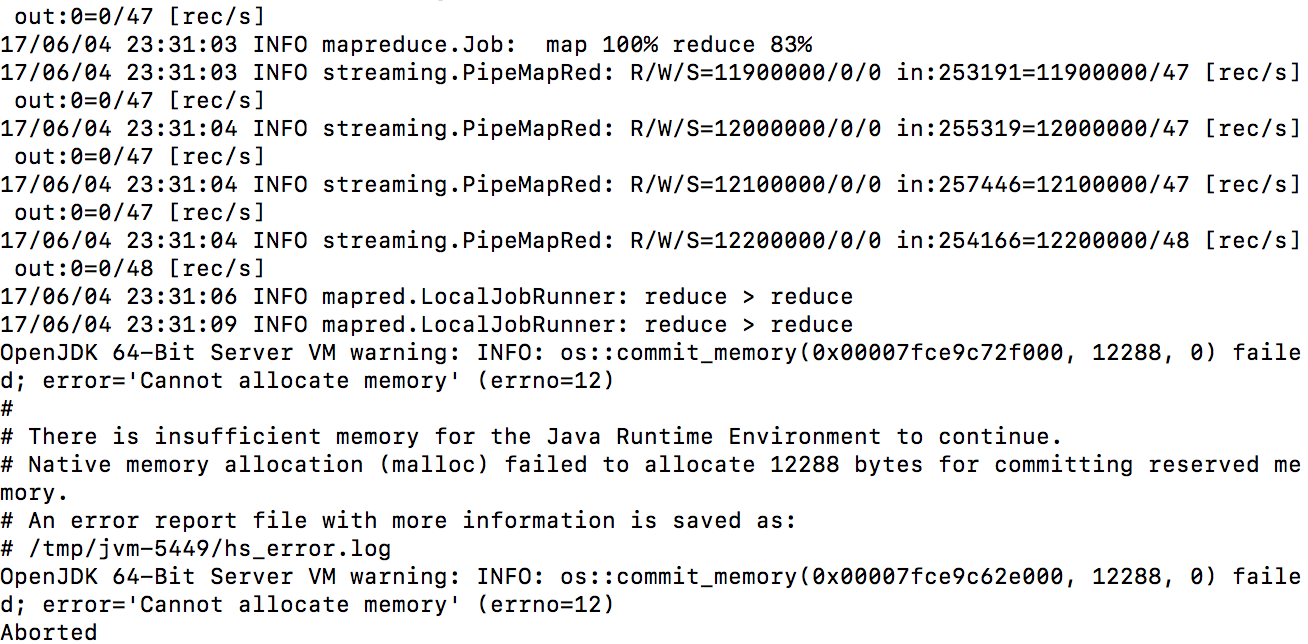
Now change the input to be the folder with all the data files:

Hadoop jar streaming:

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lo\_ss/ -output /data/kmeans\_out666 -mapper iter1Mapper.py -reducer iter1Reducer.py -file iter1Reducer.py -file iter1Mapper.py

result:

change the mapred-site.xml as instructed, but still cannot reach a result due to memory limit.

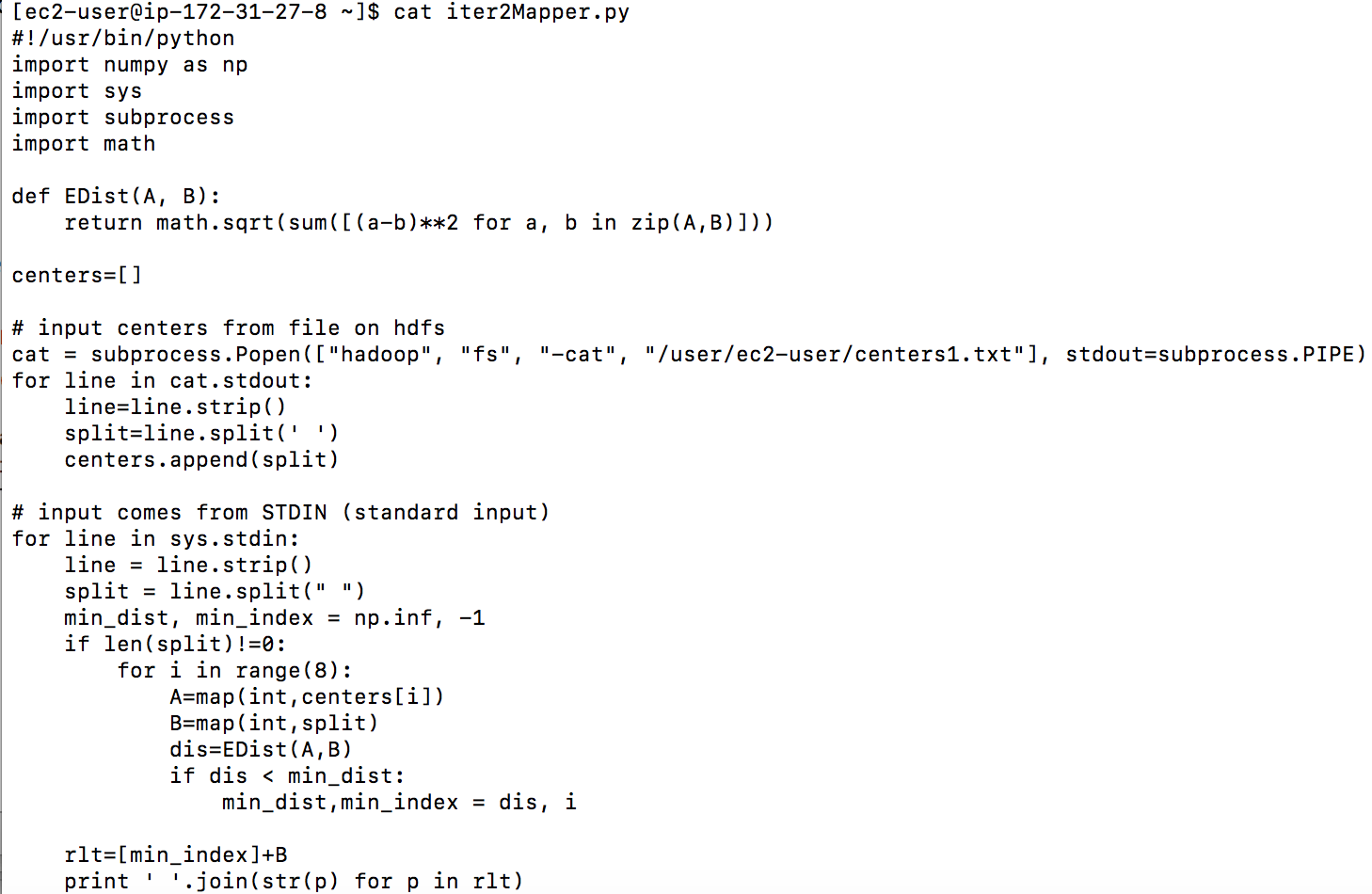


iteration 2:

replace centers with the output from last step, which is the new centers and repeat the process and gives out a new set of centers, when the centers stabilize then we could give out clustering result.

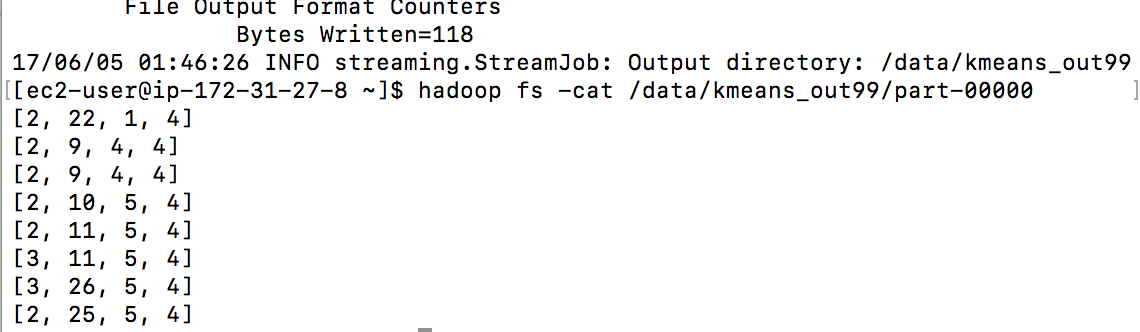
We will give a result based on the one file 000000\_0 for we had the memory problem to process all the files from last iteration.

the iter2Mapper.py is same as iter1Mapper.py except that the centers.txt is updated to be the new centers generated from iteration 1.



And the reducer code is the old one.

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lo\_ss/000000\_0 -output /data/kmeans\_out99 -mapper iter2Mapper.py -reducer iter1Reducer.py -file iter1Reducer.py -file iter2Mapper.py



# Part 4: Performance

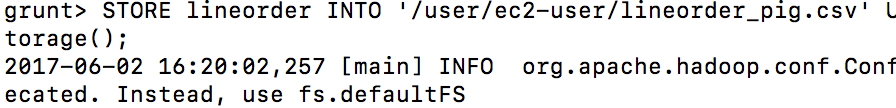
1. Both of your solutions to Part-1A # EXAPLAIN HOW THINGS SCALES BY TIME AND NUBMER OF NODES
   1. Scale4: single node and a cluster of at least 4 nodes

CLUSTER OF 4 NODES:

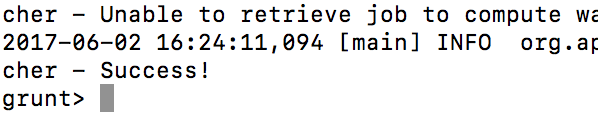
PIG:

PIG Time Taken:4 min 9 seconds

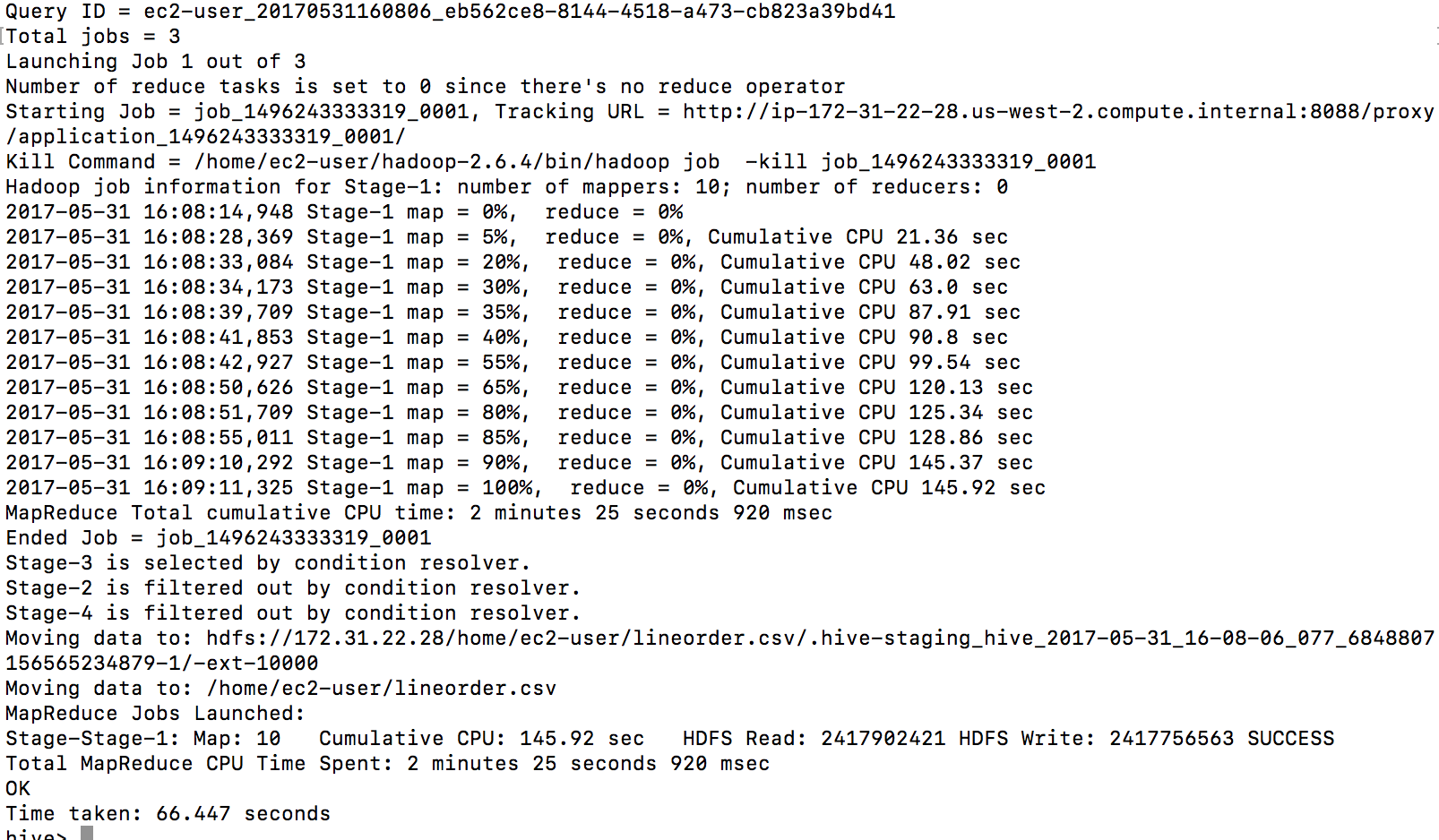
Start



End



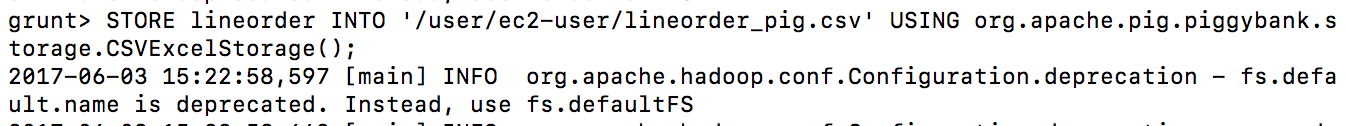
HIVE:



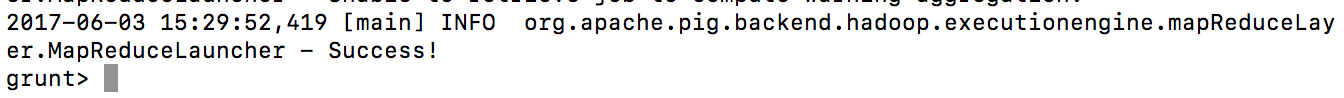
SINGLE NODE(command is same as in cluster, just using one node instead)

PIG: 6 min 54 seconds

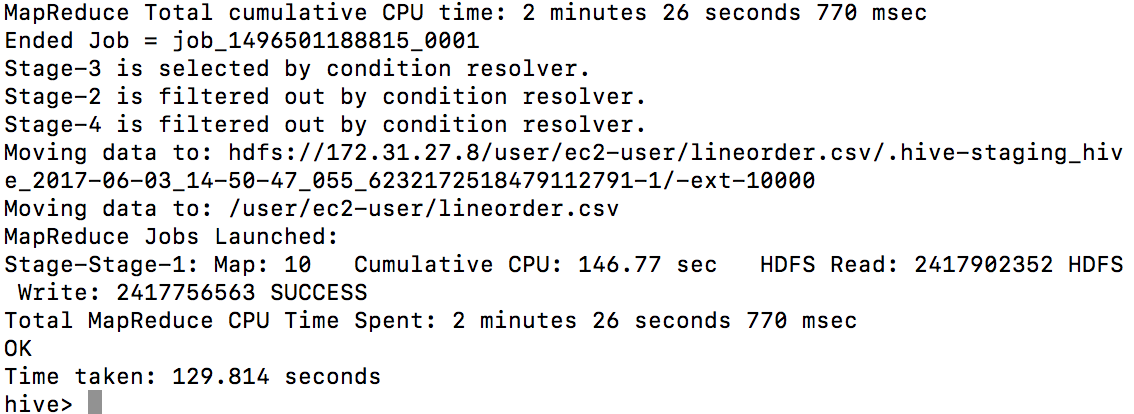
Start:



End:



HIVE:



* 1. Scale14: cluster of at least 4 nodes

HIVE:

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/Scale14/lineorder.tbl

hadoop fs -put lineorder.tbl.1 /data/lineorder\_s14.tbl

hadoop fs -ls /data # checked, everything is fine

cd $HIVE\_HOME

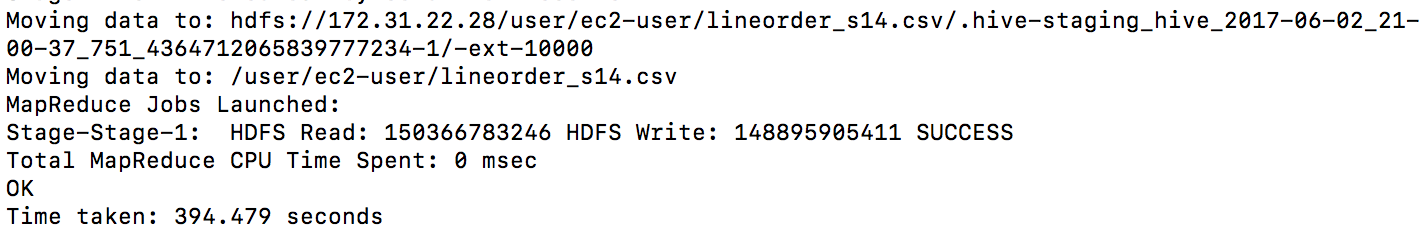
bin/home

CREATE TABLE lineorder14 ( lo\_orderkey int, lo\_linenumber int, lo\_custkey int, lo\_partkey int, lo\_suppkey int, lo\_orderdate int, lo\_orderpriority varchar(15), lo\_shippriority varchar(1), lo\_quantity int, lo\_extendedprice int, lo\_ordertotalprice int, lo\_discount int, lo\_revenue int, lo\_supplycost int, lo\_tax int, lo\_commitdate int, lo\_shipmode varchar(10) )ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA INPATH '/data/lineorder\_s14.tbl' OVERWRITE INTO TABLE lineorder14;

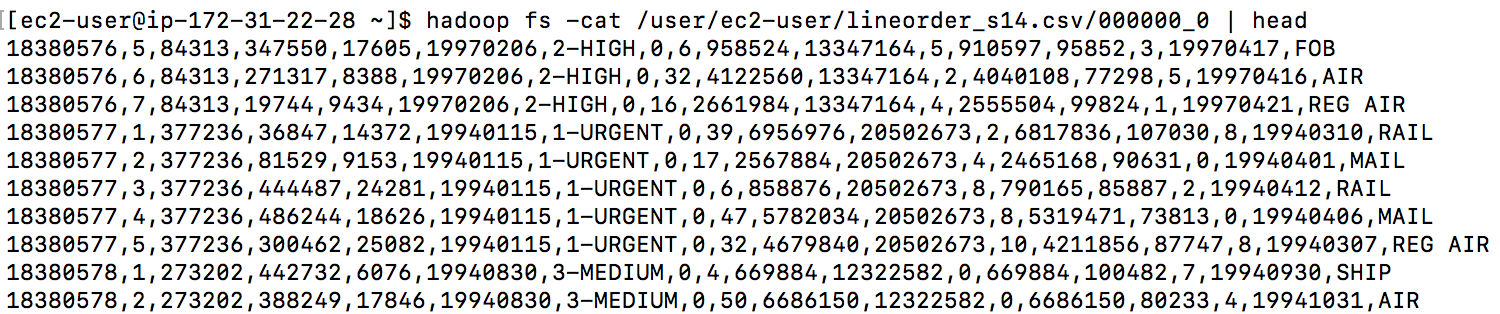
INSERT OVERWRITE DIRECTORY '/user/ec2-user/lineorder\_s14.csv' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' SELECT \* FROM lineorder14;

Time taken: 394.479 seconds



Check out result:

Hadoop fs –cat /user/ec2-user/lineorder\_s14.csv/ | head



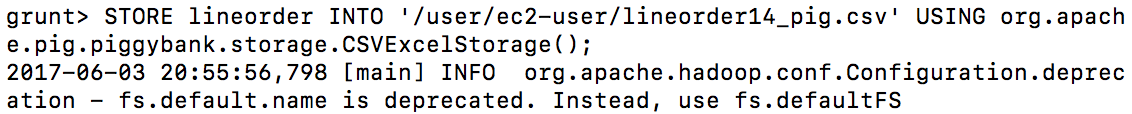
PIG:

lineorder = LOAD '/data/lineorder\_s14.tbl' USING PigStorage('|')AS (lo\_orderkey: int, lo\_linenumber: int, lo\_custkey: int, lo\_partkey: int, lo\_suppkey: int, lo\_orderdate: int, lo\_orderpriority: chararray, lo\_shippriority: chararray, lo\_quantity: int, lo\_extendedprice: int, lo\_ordertotalprice: int, lo\_discount: int, lo\_revenue: int, lo\_supplycost: int, lo\_tax: int, lo\_commitdate: int, lo\_shipmode: chararray);

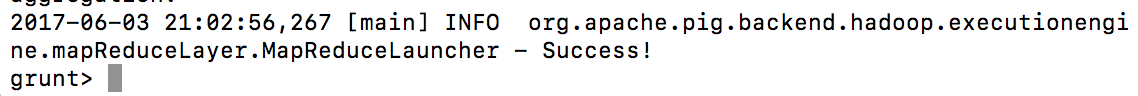
STORE lineorder INTO '/user/ec2-user/lineorder14\_pig.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage();

PIG Time Taken: 7 minutes

Start



End



B--Both of your solutions for 2-B.

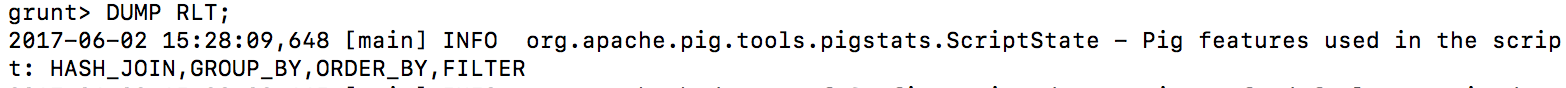
Scale4: single node and a cluster of at least 4 nodes

CLUSTER OF 4 NODES:

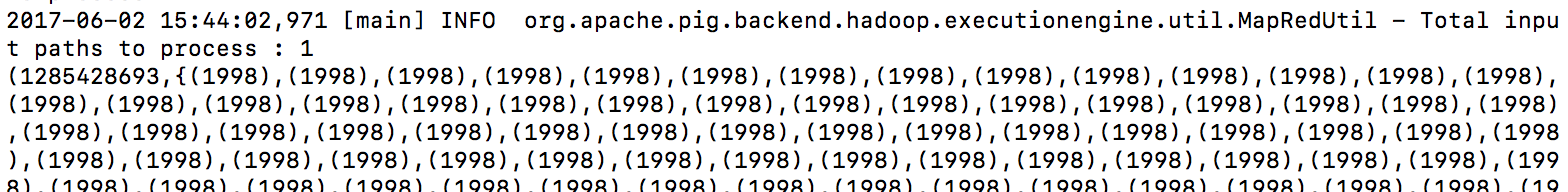
PIG:

time: 15 min 53 seconds

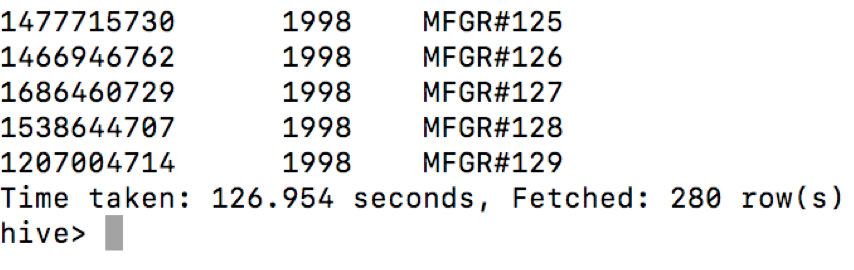
start



end

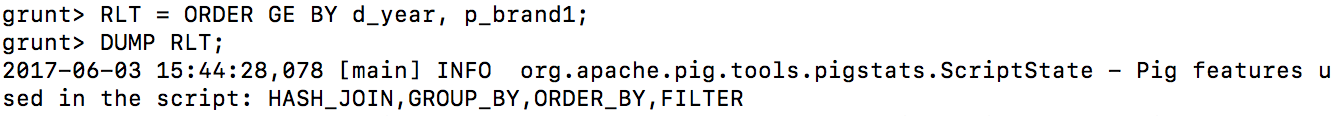


HIVE:

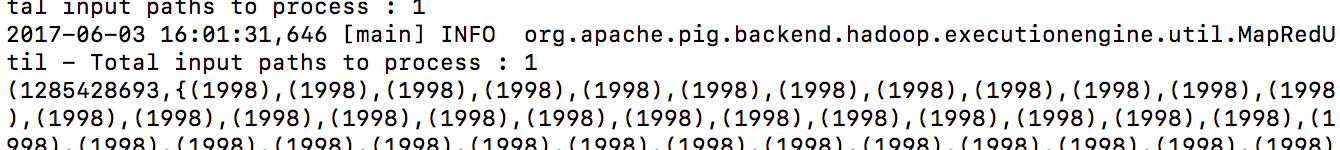


SINGLE NODE:

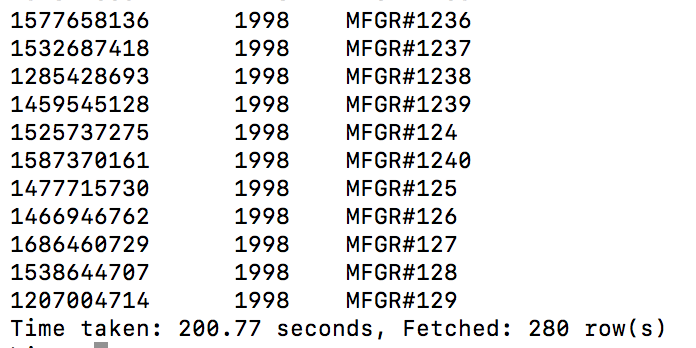
PIG:

start

end:



HIVE:



Summarize the results and cluster performance/scaling:

Pig takes more time to process queries than hive overall; one node takes more time than cluster of nodes to process tasks; when data scale up, the time it takes to process a task is much longer, as it also takes extra space and time to save and distribute the file on hdfs.