Travis Donoghue - CSC 555: Mining Big Data

Project, Phase 2

Table of Contents

**Part 1 A:** **Data Transformation 2-18**

Hive – 5 Node Cluster – Small File2-5

Hive – 5 Node Cluster – Large F**i**le6-7

Hive – Single Node Cluster8-9

PIG – Small File - 5 Node Cluster 10-11

PIG – Large File - 5 Node Cluster12-13

PIG – Large File - Single Node Cluster14-16

Hadoop Streaming - 5 Node Cluster17-18

Hadoop Streaming – Single Node Cluster17-18

**PART 1 B: Data Transformation 19-26**

PART 1 B: Hive - 5 Node Cluster19-20

PIG – Small File – 5 Node Cluster 21-22

PIG – Large File – 5 Node Cluster23-24

Hadoop Streaming – 5 Node Cluster25-26

**PART 2 A: Querying 27-35**

Downloading Data27-28

Tables29-30

Load Data 31-32

HIVE Queries33-35

**Part 2 B: Querying 36-47**

HIVE – 5 Node Cluster36-38

HIVE – Single Node Cluster39-40

Pig – 5 Node Cluster41-43

Pig – Single Node Clsuter44-46

**Part 3: Clustering**47-51

Mahout Sythetic Clustering - KMeans47-50

Hadoop KMeans Clustering51

**Part 4: Performance52-53**

**EXTRA CREDIT54-55**

In this part of the project, you will various queries using Hive, Pig and Hadoop streaming. The schema is available below, but don’t forget to apply the correct delimiter:

http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM\_schema\_hive.sql

http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/ (this is Scale4)

<http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/Scale14/>

In your submission, please note what instance and what cluster you are using (you can reuse your existing cluster for most of the questions). Please be sure to submit all code (pig, python and Hive). You should also submit the command lines you use and a screenshot of a completed run (just the last page, do not worry about capturing the whole output). An answer submission with screenshot/results but without the code will not receive credit.

I highly recommend creating a small sample input (e.g., by running head lineorder.tbl > lineorder.tbl.sample and testing your code with it, you can use head -n 100 to get first 100 lines).

**Part 1: Data Transformation**

1. **Transform lineorder.tbl table into a comma-separated file: Use Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions)**

**PART 1 A: HIVE – 5 Node Cluster**

cd $HIVE\_HOME

bin/hive

If you need to restart Hive because you are unable to get into it. Run this command to clean all data. Be sure you are in apache-hive-2.01.1

stop-all.sh

rm -rf /tmp/hadoop-ec2-user/dfs/

go to each node by ssh “private IP Address” and run the same command

rm -rf /tmp/hadoop-ec2-user/dfs

then you format

hadoop namenode -format

Restart

start-dfs.sh

start-yarn.sh

mr-jobhistory-daemon.sh start historyserver

hadoop dfsadmin -report

Clear out hive (start fresh) – you will get an error when loading data into the table if you do not

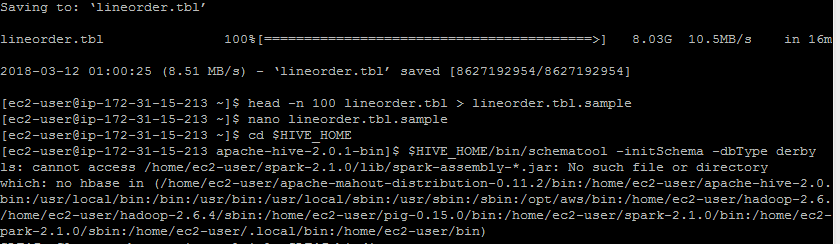
rm -rf metastore\_db/

$HIVE\_HOME/bin/schematool -initSchema -dbType derby

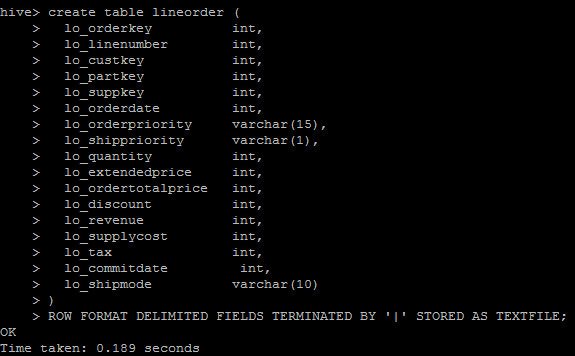
hadoop dfsadmin -safemode leave

**Downloaded the file lineorder.tbl**

**Converted the large linorder.tbl file to smaller 100 row file called “lineorder.tbl.sample”**



**Created table lineorder expecting input of ‘|’ delimited file**



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,

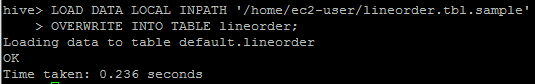
lo\_commitdate int,

lo\_shipmode varchar(10)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

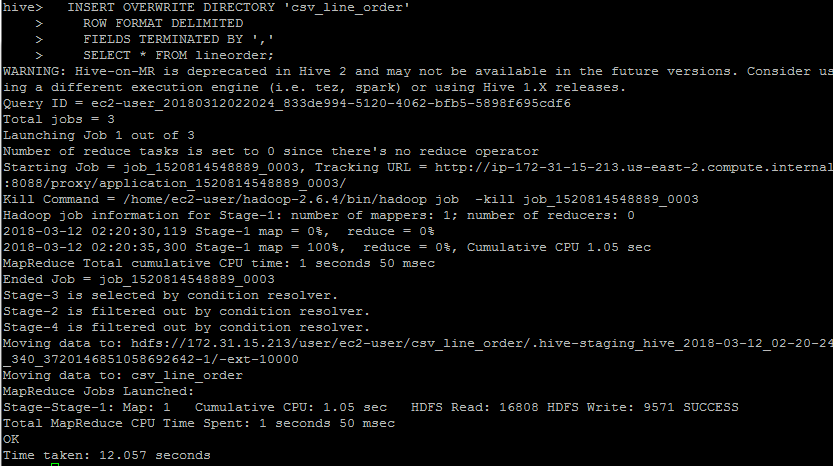
**Loaded smaller file lineorder.tbl.sample file of 100 records into the table**



LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl.sample'

OVERWRITE INTO TABLE lineorder;

**Using the smaller file lineorder.tbl.sample within the table linorder; exporting all contents of the table into a .csv file in HDFS user home directory labeled csv\_line\_order (/user/ec2-user/csv\_line\_order)**



INSERT OVERWRITE DIRECTORY 'csv\_line\_order'

ROW FORMAT DELIMITED

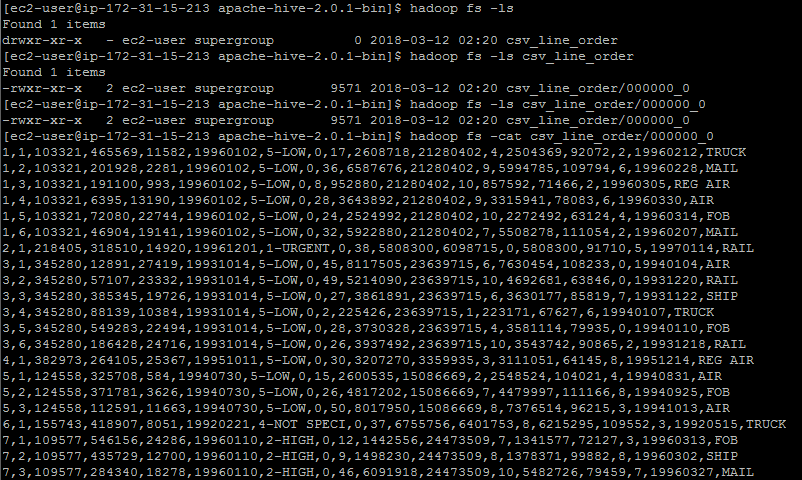
FIELDS TERMINATED BY ','

SELECT \* FROM lineorder;

hadoop fs -ls Command to see what files are located in HDFS

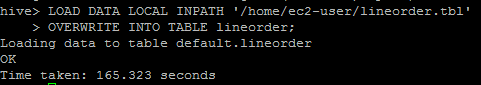
hadoop fs -ls csv\_line\_order Command to see what files are located in csv\_line\_order

hadoop fs -cat csv\_line\_order/000000\_0 Command to see what the file looks like



**PART 1 A: HIVE – 5 Node Cluster – Large File**

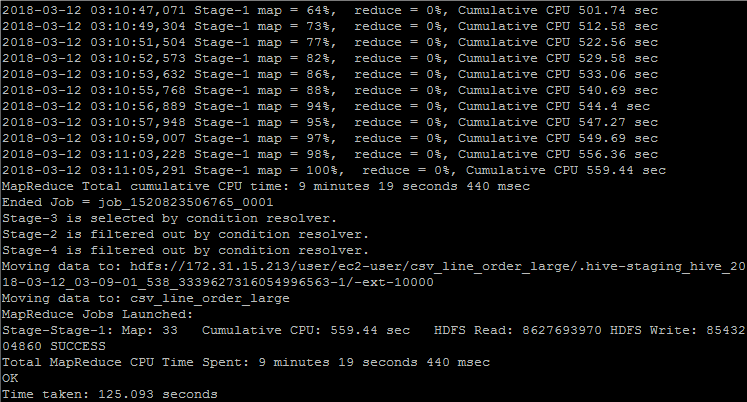
**Loaded entire file lineorder.tbl file of 100 records into the table**



LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl'

OVERWRITE INTO TABLE lineorder;

**Using the larger file lineorder.tbl within the table linorder; exporting all contents of the table into a .csv file in HDFS user home directory labeled csv\_line\_order\_large (/user/ec2-user/csv\_line\_order\_large)**



INSERT OVERWRITE DIRECTORY 'csv\_line\_order\_large'

ROW FORMAT DELIMITED

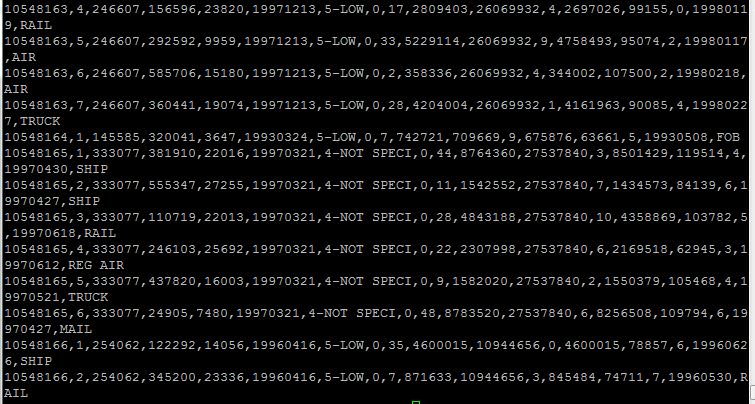
FIELDS TERMINATED BY ','

SELECT \* FROM lineorder;

hadoop fs -ls Command to see what files are located in HDFS

hadoop fs -ls csv\_line\_order\_large Command to see what files are located in csv\_line\_order

hadoop fs -cat csv\_line\_order\_large/000000\_0 Command to see what the file looks like



INSERT OVERWRITE DIRECTORY 'csv\_line\_order\_large\_singlenode'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT \* FROM lineorder;

**PART 1 A: HIVE – Single Node Cluster – Large File**

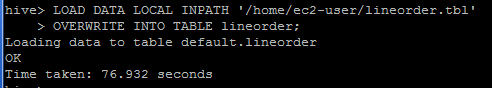
cd $HIVE\_HOME

bin/hive

**Loaded entire file lineorder.tbl file**

LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl'

OVERWRITE INTO TABLE lineorder;



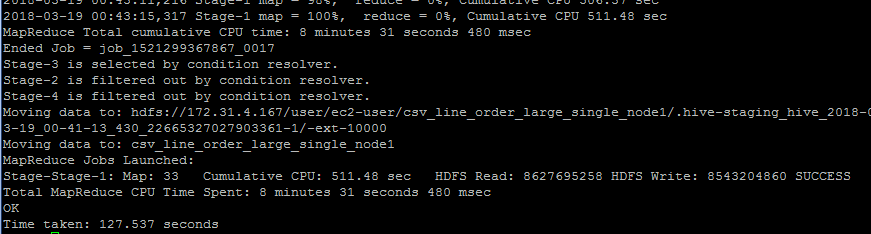
**Using the larger file lineorder.tbl within the table lineorder; exporting all contents of the table into a .csv file in HDFS user home directory labeled csv\_line\_order\_large (/user/ec2-user/csv\_line\_order\_large)**

INSERT OVERWRITE DIRECTORY 'csv\_line\_order\_large\_single\_node1'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

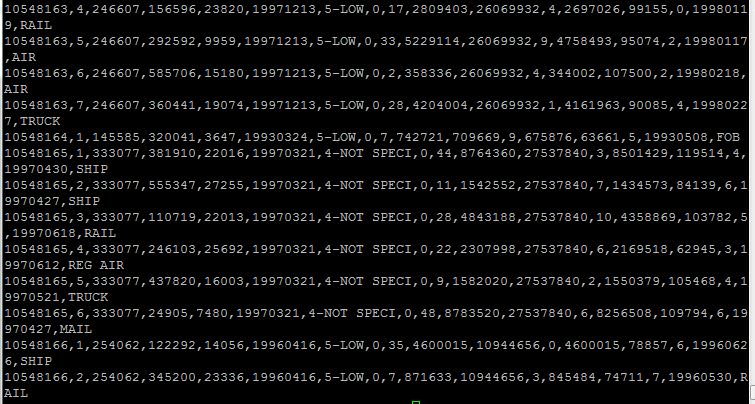
SELECT \* FROM lineorder;



hadoop fs -ls Command to see what files are located in HDFS

hadoop fs -ls csv\_line\_order\_large\_single\_node1 Command to see what files are located in csv\_line\_order

hadoop fs -cat csv\_line\_order\_large\_single\_node1/000000\_0 Command to see what the file looks like



**PART 1 A: PIG - SMALL FILE - 5 Node Cluster**

cd $PIG\_HOME

bin/pig

**To insure the file exists**

hadoop fs -mkdir /user/ec2-user

**To place the file where PIG can retrieve it**

hadoop fs -put ../lineorder.tbl.sample /user/ec2-user/

**To verify the file is there**

hadoop fs -ls /user/ec2-user/lineorder.tbl.sample

**Go back into PIG**

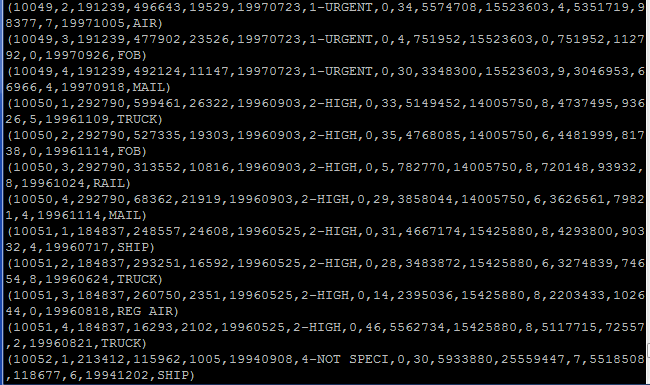
bin/pig

**Load the File**

lineordersmall = LOAD 'lineorder.tbl.sample' 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);

**View the file**

Dump lineordersmall;

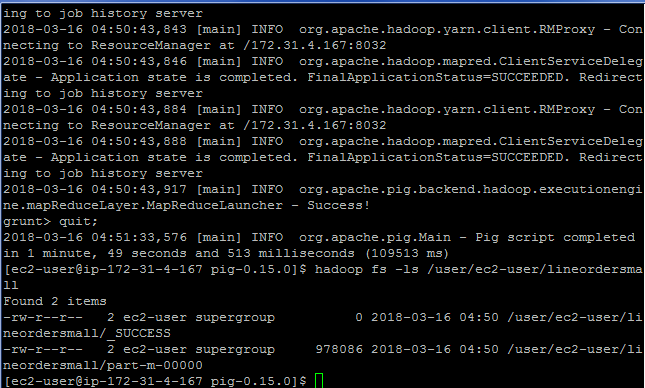


**To transform and store the file as a comma separated file**

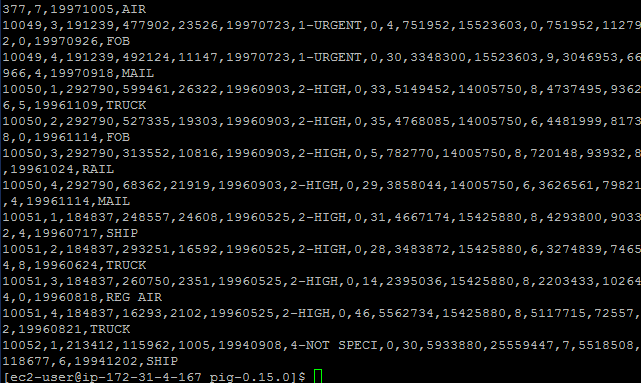
Store lineordersmall INTO 'lineordersmall' USING PigStorage (',');

**To verify the file is there**

hadoop fs -ls /user/ec2-user/lineordersmall



hadoop fs -cat lineordersmall/part-m-00000 Command to see what the file looks like



**PART 1 A: PIG - LARGE FILE - 5 Node Cluster**

**To insure the -mkdir file exists**

hadoop fs -mkdir /user/ec2-user

**To place the file where PIG can retrieve it**

hadoop fs -put ../lineorder.tbl /user/ec2-user/

**To verify the file is there**

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

**Go back into PIG**

cd $PIG\_HOME

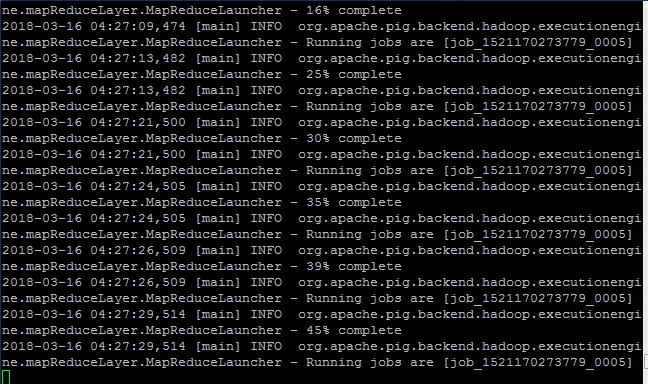
bin/pig

**Load the File**

lineorderlarge = LOAD 'lineorder.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);

**View the file (TOO LARGE TO DUMP)**

~~Dump lineorder.tbl;~~



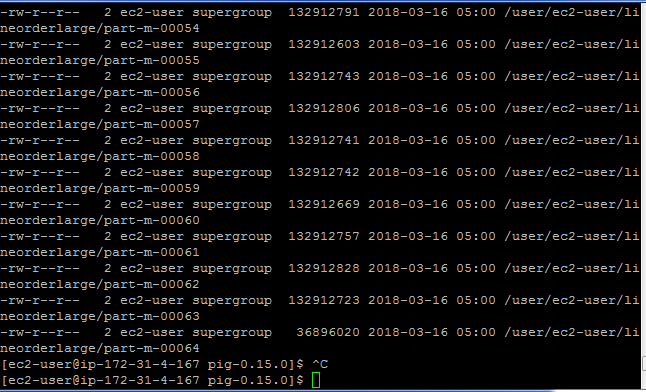
**To transform and store the file as a comma separated file**

Describe lineorderlarge; Store lineorderlarge INTO 'lineorderlarge2' USING PigStorage (',');

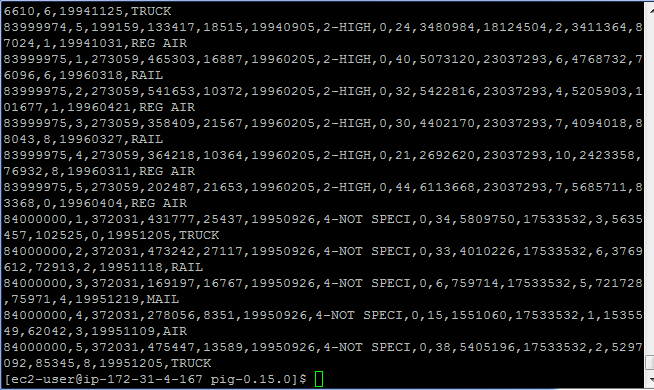


**To verify the file is there**

hadoop fs -ls /user/ec2-user/lineorderlarge **Note:** **There are 64 files that make up lineorder**



hadoop fs -cat lineorderlarge/part-m-00064 **Command to see what the file looks like (only pulled the last file)**



**PART 1 A: PIG - LARGE FILE - Single Node Cluster**

**To insure the -mkdir file exists**

hadoop fs -mkdir /user/ec2-user

**To place the file where PIG can retrieve it**

hadoop fs -put ../lineorder.tbl /user/ec2-user/

**To verify the file is there**

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

**Go back into PIG**

cd $PIG\_HOME

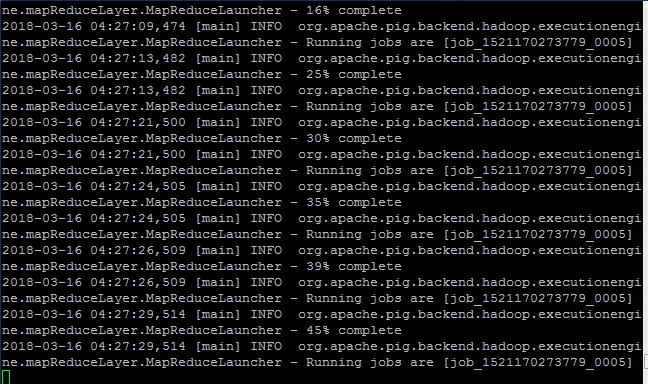
bin/pig

**Load the File**

Lineorderlarge\_single\_node = LOAD 'lineorder.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);

**View the file (TOO LARGE TO DUMP)**

~~Dump lineorder.tbl;~~



**To transform and store the file as a comma separated file**

Describe Lineorderlarge\_single\_node; Store Lineorderlarge\_single\_node INTO 'Lineorderlarge\_single\_node1' USING PigStorage (',');

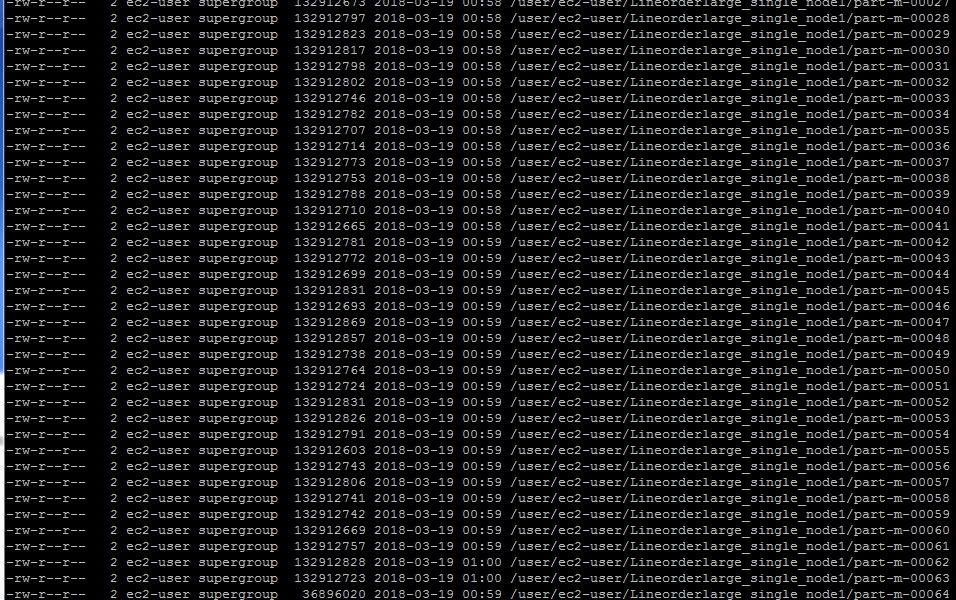


**Used the StartAt and FinshedAt to determine the time it took**

**To verify the file is there**

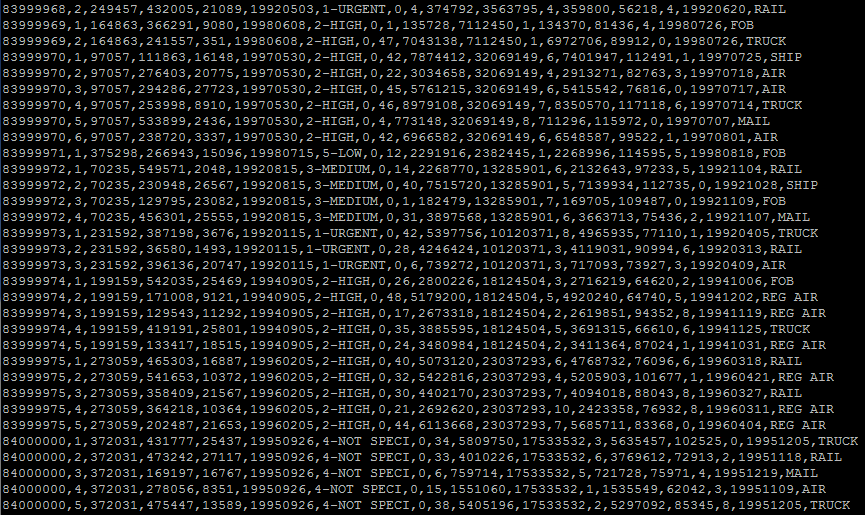
hadoop fs -ls /user/ec2-user/Lineorderlarge\_single\_node1

**Note:** **There are 64 files that make up lineorder**



hadoop fs -cat Lineorderlarge\_single\_node1/part-m-00064

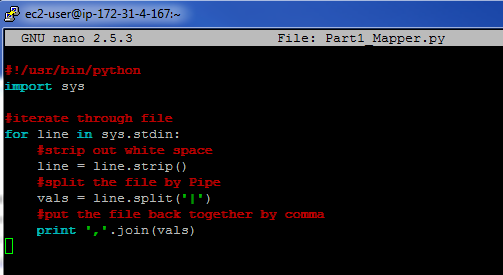
**Command to see what the file looks like (only pulled the last file)**



**PART 1 A: Hadoop Streaming - 5 Node & Single Node**

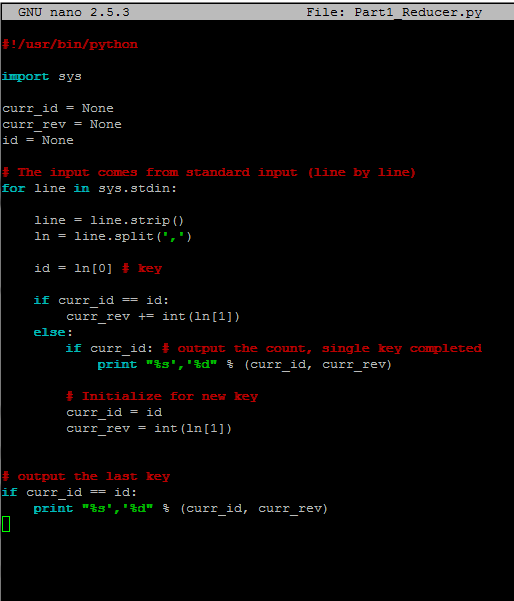
**Write the mapper code**

nano Part1\_Mapper.py



**Write the reducer code**

nano Part1\_Reducer.py



**lineorder.tbl.sample**

**Create a destination directory**

hadoop fs -mkdir /data

**Copy the file to HDFS for processing (if needed)**

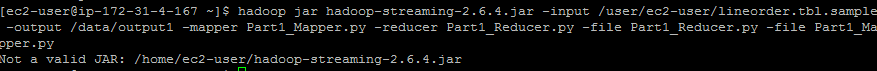
hadoop fs -put lineorder.tbl.sample /data/

**Verify the file was uploaded**

hadoop fs -ls /data

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lineorder.tbl -output /data/output1 -mapper Part1\_Mapper.py -reducer Part1\_Reducer.py -file Part1\_Reducer.py -file Part1\_Mapper.py

**Kept receiving the error “Not a valid JAR”. I ran the movie lens without any issues.**



hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/mlens -output /data/output4 -mapper myMapper.py -reducer myReducer.py -file myReducer.py -file myMapper.py

**PART 1 B: HIVE - 5 Node Cluster**

cd $PIG\_HOME

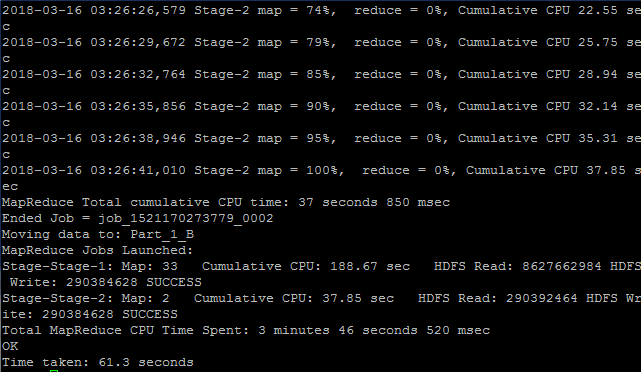
bin/pig

INSERT OVERWRITE DIRECTORY 'Part\_1\_B'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ' '

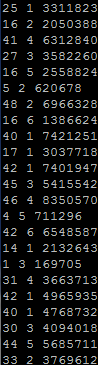
SELECT lo\_quantity, lo\_linenumber, lo\_revenue FROM lineorder WHERE lo\_discount BETWEEN 6 and 8;



hadoop fs -ls Command to see what files are located in HDFS

hadoop fs -ls Part\_1\_B Command to see what files are located in csv\_line\_order

hadoop fs -cat Part\_1\_B/000001\_0 Command to see what the file looks like



**PART 1 B: PIG – SMALL FILE - 5 Node Cluster**

cd $PIG\_HOME

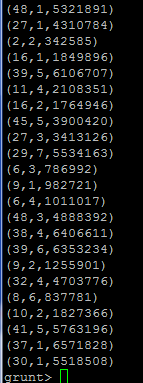
bin/pig

First = FILTER lineordersmall BY lo\_discount > 6;

Second = FILTER First BY lo\_discount < 8;

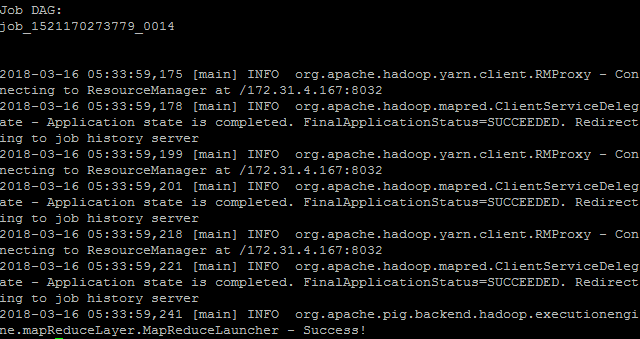
Third = FOREACH Second GENERATE lo\_quantity, lo\_linenumber, lo\_revenue;

Dump Third;



**To transform and store the file as a space separated file**

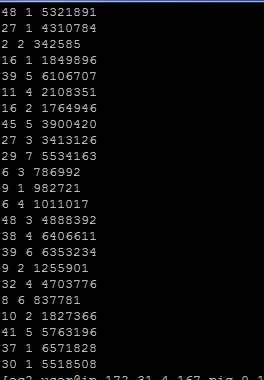
Store Third INTO 'Third\_small' USING PigStorage (' ');



**To verify the file is there**

hadoop fs -ls /user/ec2-user/Third\_small **Note:** **There is 1 that makes up Third\_small**

hadoop fs -cat Third\_small/part-m-00000 **Command to see what the file looks like**



**PART 1 B: PIG – LARGE FILE - 5 Node Cluster**

cd $PIG\_HOME

bin/pig

First\_large = FILTER lineorderlarge BY lo\_discount > 6;

Second\_large = FILTER First\_large BY lo\_discount < 8;

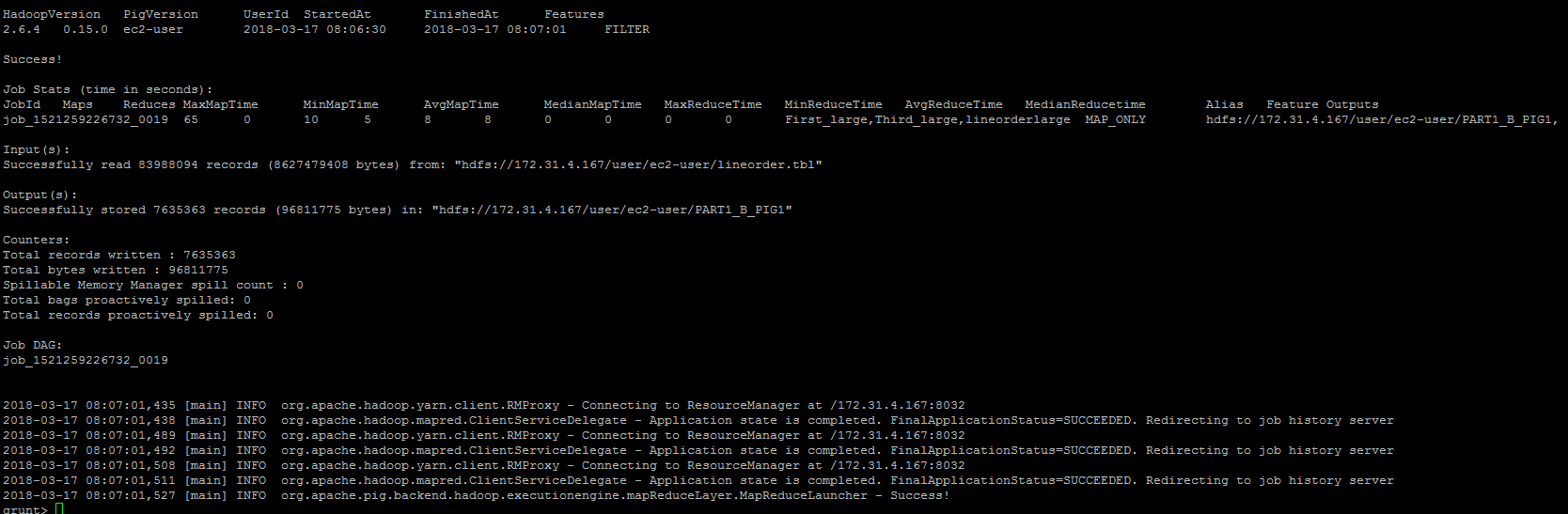
Third\_large = FOREACH Second\_large GENERATE lo\_quantity, lo\_linenumber, lo\_revenue;

T = LIMIT Third\_large 10;

Describe T; Dump T;

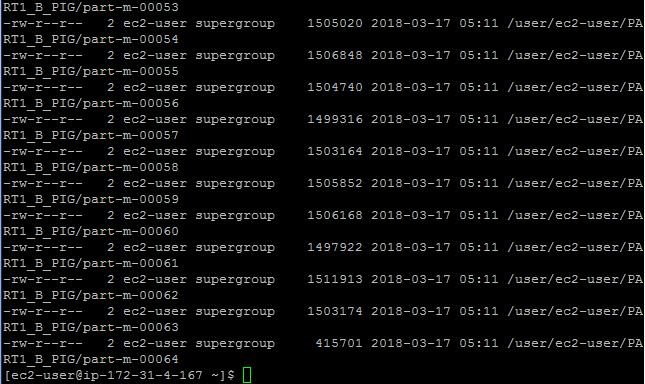
**To transform and store the file as a space separated file**

Describe Third\_large; Store Third\_large INTO 'PART1\_B\_PIG' USING PigStorage (' ');

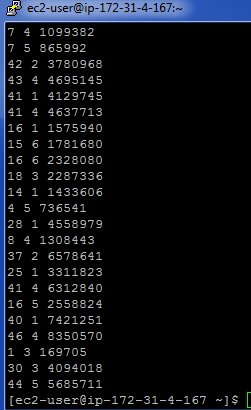


**To verify the file is there (leave pig first)**

hadoop fs -ls /user/ec2-user/PART1\_B\_PIG **Note:** **There are 64 files that make up Third\_large**



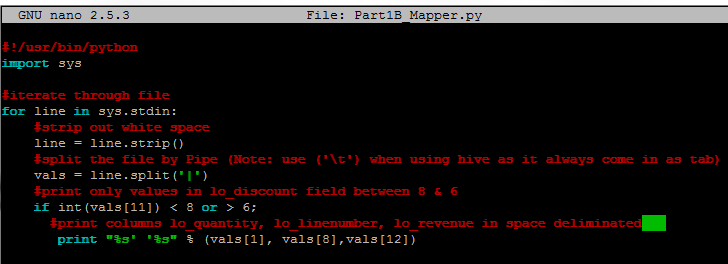
hadoop fs -cat /user/ec2-user/PART1\_B\_PIG/part-m-00064 **Command to see what the file looks like**



**PART 1 B: Hadoop Streaming – 5 Node**

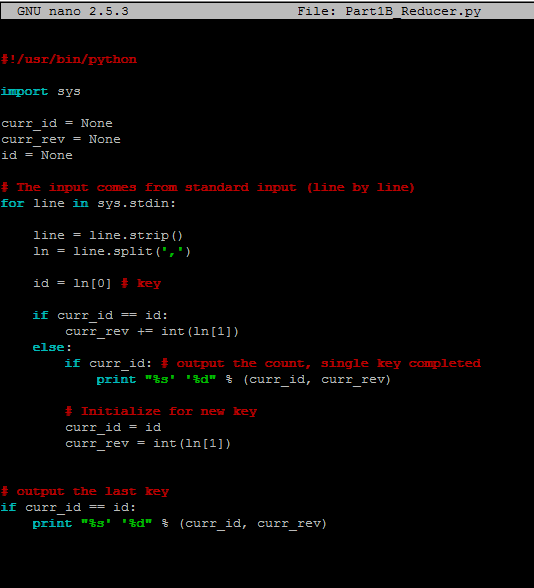
**Write the mapper code**

nano Part1B\_Mapper.py



**Write the Reducer code**

nano Part1B\_Reducer.py



hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lineorder.tbl -output /data/output1 -mapper Part1B\_Mapper.py -reducer Part1B\_Reducer.py -file Part1\_Reducer.py -file Part1\_Mapper.py

**Kept receiving the error “Not a valid JAR”. I spent many hours on this, and as troubleshooting technique, I re-ran the movie lens without any issues. I am really looking forward to knowing what the answer to this one is, or maybe identify what I did wrong. I think the reducer code is part of the problem. Appreciate any knowledge you can provide. Thank you.**

**B. Extract three of the numeric columns (lo\_quantity, lo\_linenumber, lo\_revenue) for rows where lo\_discount is between 6 and 8 into a space-separated text file (for K-Means clustering later). Use Hive, MapReduce with Hadoop Streaming, and Pig (3 different solutions)**

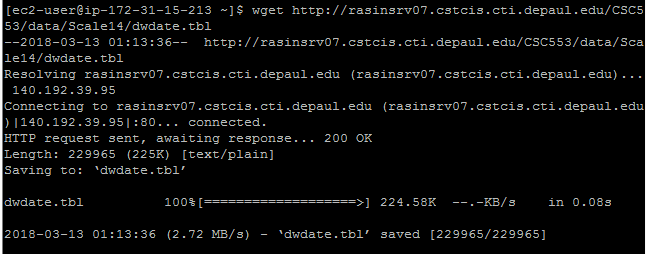
**Part 2: Querying**

All queries from SSBM benchmark are available here:

http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM\_queries\_all.sql

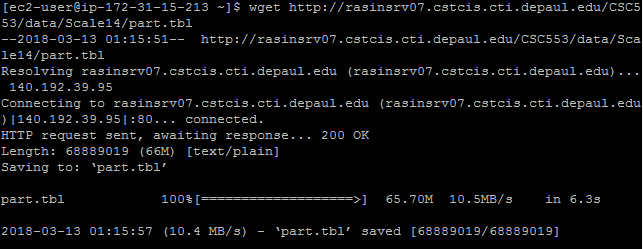
Using Scale4 data perform the following data processing and don’t forget to time your results.

**Download dwdate.tbl data**



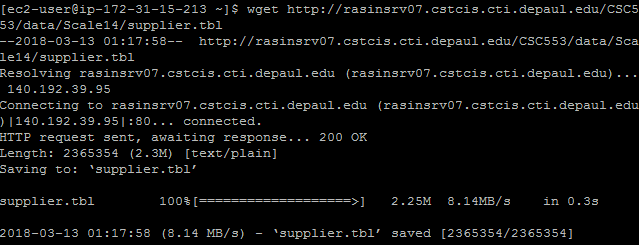
wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/Scale14/dwdate.tbl>

**Download part.tbl data**



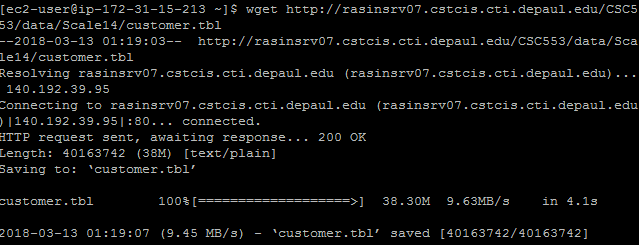
wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/Scale14/part.tbl>

**Download supplier.tbl data**



wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC553/data/Scale14/supplier.tbl>

**Download customer.tbl data**



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

**Create Tables part, supplier, customer, dwdate**

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 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;

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;

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;

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,

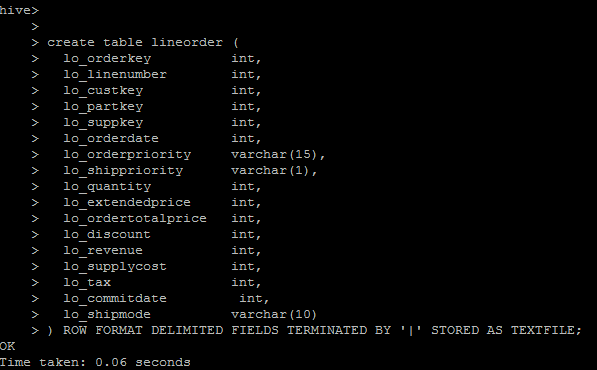
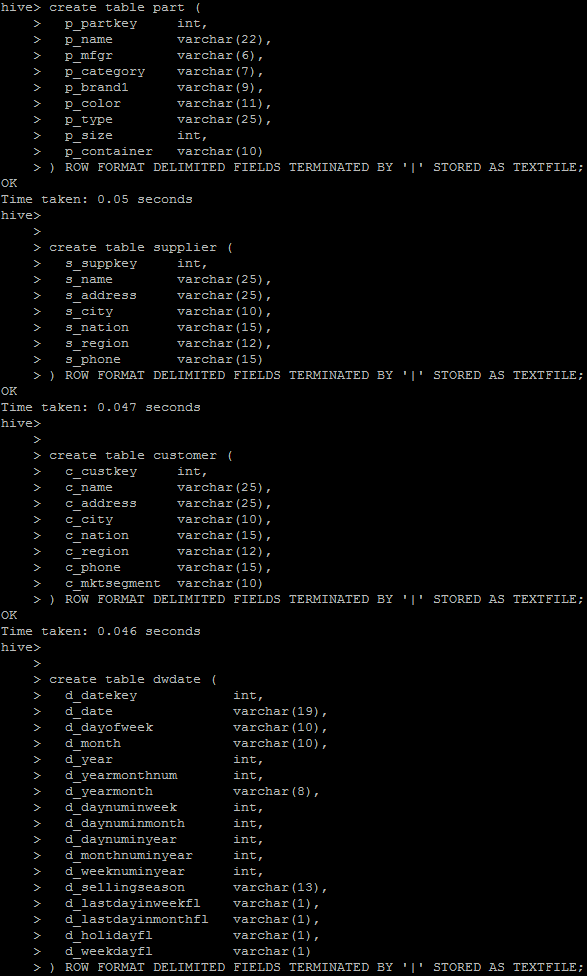
lo\_supplycost int,

lo\_tax int,

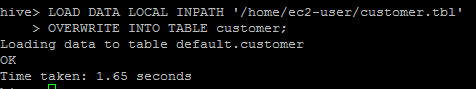
lo\_commitdate int,

lo\_shipmode varchar(10)

) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;



**Load customer.tbl data in customer table**



**LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl'**

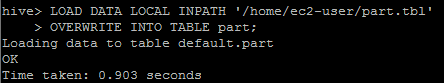
**OVERWRITE INTO TABLE customer;**

**Check to see if there is data**

**Select \* FROM customer limit 5;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Load part.tbl data in part table**



**LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl'**

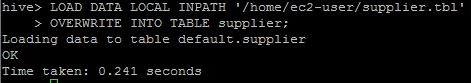
**OVERWRITE INTO TABLE part;**

**Check to see if there is data**

**Select \* FROM customer limit 5;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Load supplier.tbl data in supplier table**



**LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl'**

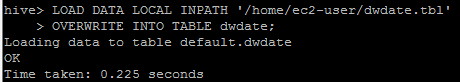
**OVERWRITE INTO TABLE supplier;**

**Check to see if there is data**

**Select \* FROM supplier limit 5;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Load dwdate.tbl data in dwdate table**



**LOAD DATA LOCAL INPATH '/home/ec2-user/dwdate.tbl'**

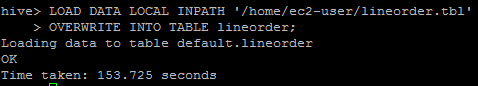
**OVERWRITE INTO TABLE dwdate;**

**Check to see if there is data**

**Select \* FROM dwdate limit 5;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Load lineorder.tbl data in lineorder table**



**LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl'**

**OVERWRITE INTO TABLE lineorder;**

**Check to see if there is data**

**Select \* FROM lineorder limit 5;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PART 2 A: HIVE**

1. Run SSBM queries 2.1, 3.3 and 4.3 using Hive only (if you have issues running the queries, try placing lineorder table first in the FROM clause of the query)

http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM\_queries\_all.sql

**Run Query 2.1**

--Q2.1 No simpifications

select sum(lo\_revenue), d\_year, p\_brand1

from lineorder, dwdate, part, supplier

where lo\_orderdate = d\_datekey

and lo\_partkey = p\_partkey

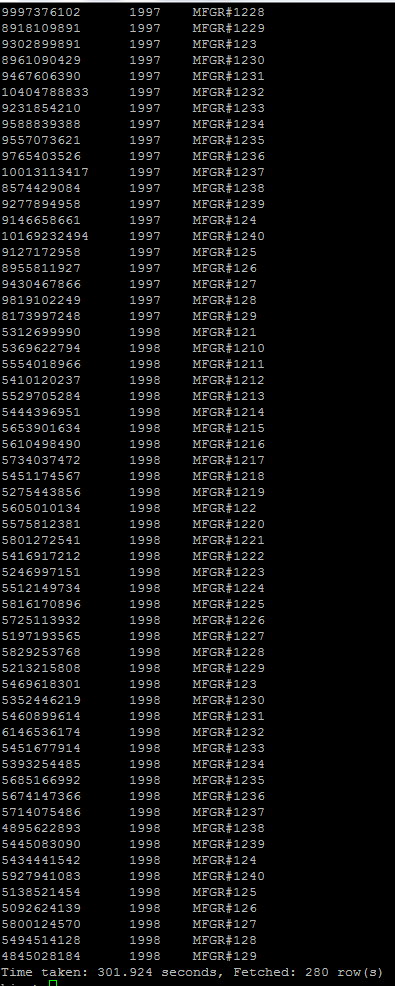
and lo\_suppkey = s\_suppkey

and p\_category = 'MFGR#12'

and s\_region = 'AMERICA'

group by d\_year, p\_brand1

order by d\_year, p\_brand1;



**Run Query 3.3**

select c\_city, s\_city, 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\_nation = 'UNITED KINGDOM'

and c\_city='UNITED KI1'

and s\_city='UNITED KI1'

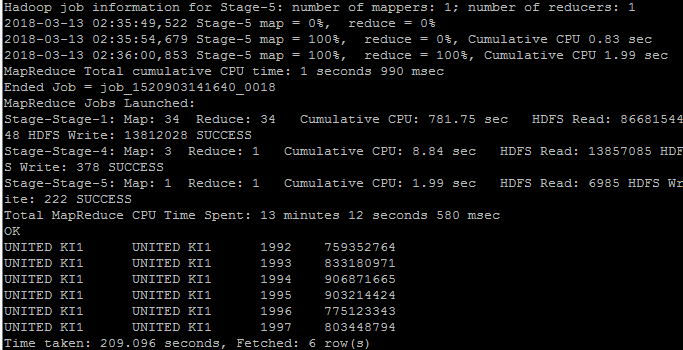
and s\_nation = 'UNITED KINGDOM'

and d\_year between 1992 and 1997

-- and d\_year >= 1992 and d\_year <= 1997

group by c\_city, s\_city, d\_year

order by d\_year asc, revenue asc;



**Run Query 4.3**

select d\_year, s\_city, p\_brand1, 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'

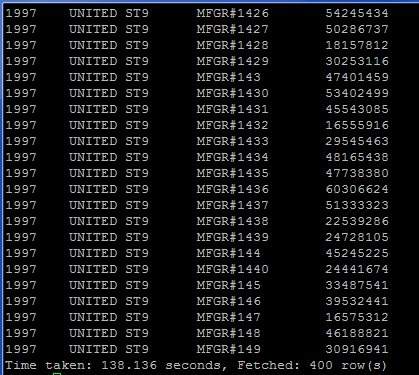
and s\_nation = 'UNITED STATES'

and d\_year = 1997

and p\_category = 'MFGR#14'

group by d\_year, s\_city, p\_brand1;

**WHY DID THIS QUERY ONLY WORK WHEN lineorder WAS IN THE BEGINNING? This confused me so much and I lost hours on it…. Please Please explain…**



**PART 2 B: HIVE – 5 Node Cluster**

Create a pre-join (i.e. a new data file) that corresponds to the following query below. You can think of it as a materialized view. What is the size of the new file? Use Hive and Pig (2 different solutions and be sure to report the file size for both).

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, lo\_supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

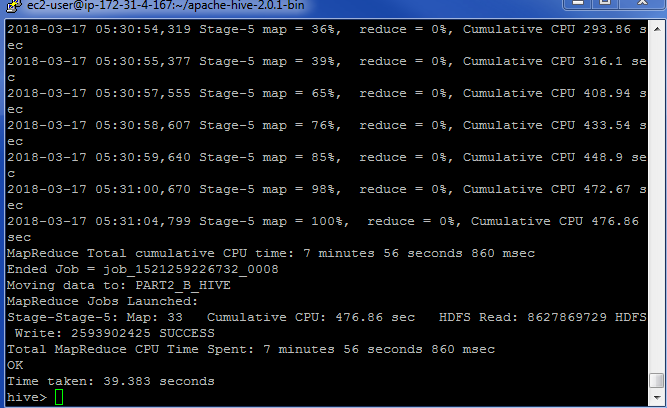
INSERT OVERWRITE DIRECTORY 'PART2\_B\_HIVE'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

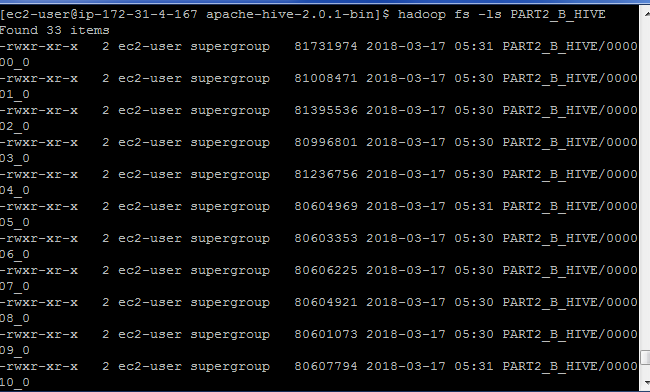
FROM lineorder, dwdate, supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

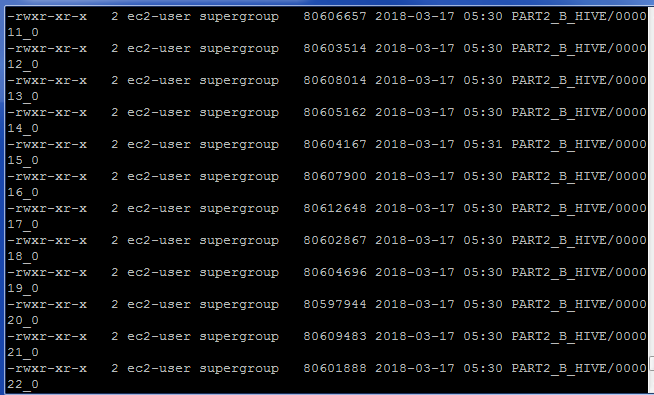


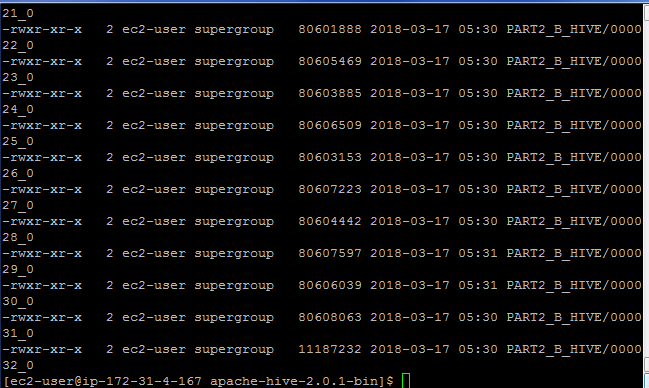
hadoop fs -ls Command to see what files are located in HDFS

hadoop fs -ls PART2\_B\_HIVE

Command to see what files are located in PART2\_B\_HIVE and how large the files are (33 Files)

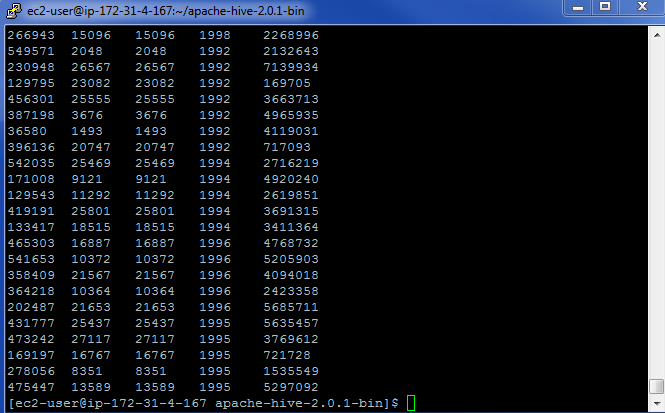






Complete file(s) sizes total = 2,674,504,313

hadoop fs -cat PART2\_B\_HIVE/000032\_0 Command to see what the file looks like (file 32)



**PART 2 B: HIVE – Single Node Cluster**

Create a pre-join (i.e. a new data file) that corresponds to the following query below. You can think of it as a materialized view. What is the size of the new file? Use Hive and Pig (2 different solutions and be sure to report the file size for both).

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, lo\_supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

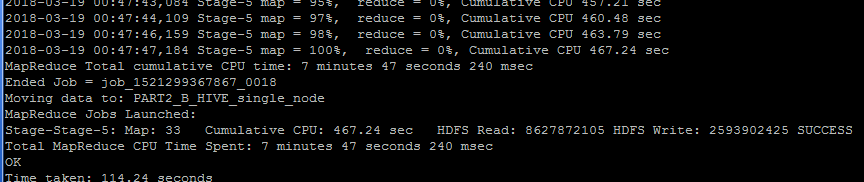
INSERT OVERWRITE DIRECTORY 'PART2\_B\_HIVE\_single\_node'

ROW FORMAT DELIMITED

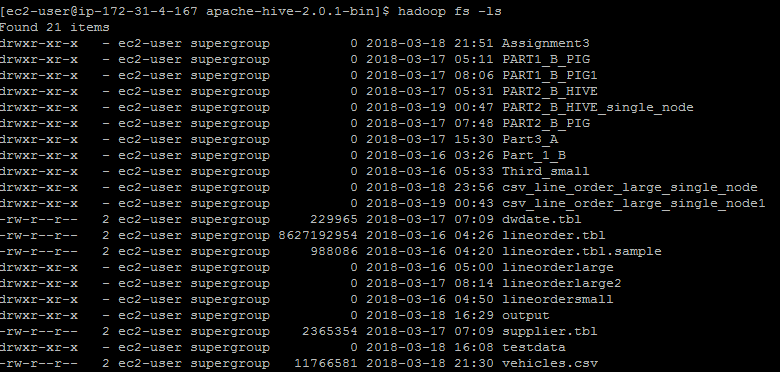
FIELDS TERMINATED BY '\t'

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

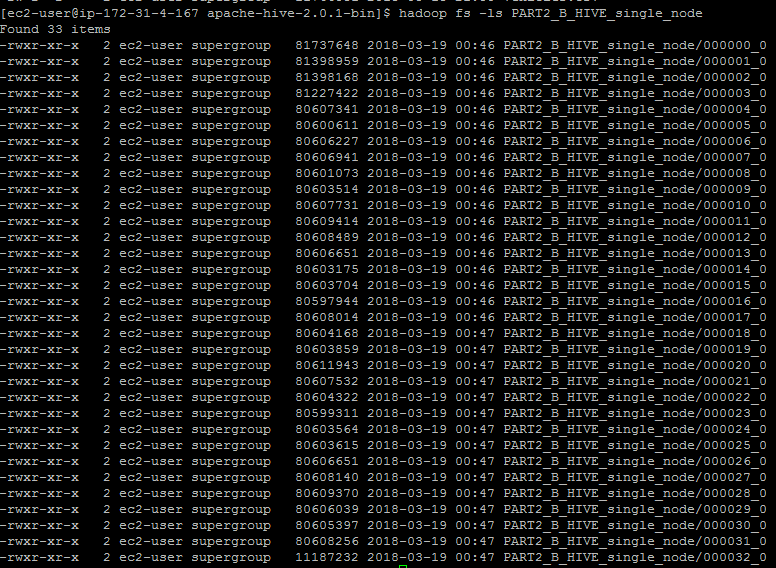


hadoop fs -ls Command to see what files are located in HDFS



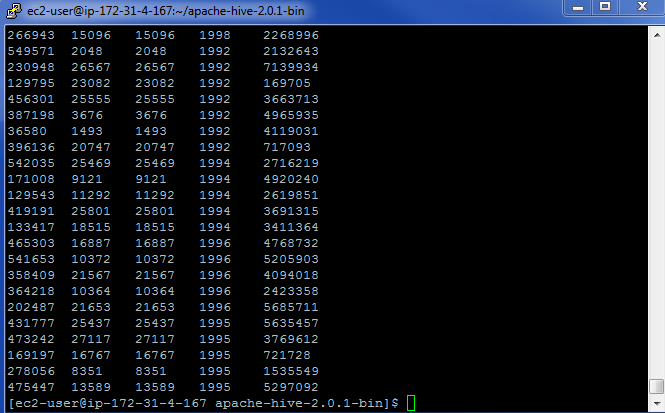
hadoop fs -ls PART2\_B\_HIVE\_single\_node

Command to see what files are located in PART2\_B\_HIVE\_single\_node and how large the files are (33 Files)



Complete file(s) sizes total = 2,674,504,313

hadoop fs -cat PART2\_B\_HIVE\_single\_node /000032\_0 Command to see what the file looks like (file 32)



**PART 2 B: PIG – 5 Node Cluster**

Create a pre-join (i.e. a new data file) that corresponds to the following query below. You can think of it as a materialized view. What is the size of the new file? Use Hive and Pig (2 different solutions and be sure to report the file size for both).

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, lo\_supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

INSERT OVERWRITE DIRECTORY 'hive\_pre-join\_file'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, supplier

WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

cd $PIG\_HOME

bin/pig

**To insure the -mkdir file exists**

hadoop fs -mkdir /user/ec2-user

**To place the lineorder file where PIG can retrieve it**

hadoop fs -put ../lineorder.tbl /user/ec2-user/

**To place the dwdate file where PIG can retrieve it**

hadoop fs -put ../dwdate.tbl /user/ec2-user/

**To place the supplier file where PIG can retrieve it**

hadoop fs -put ../supplier.tbl /user/ec2-user/

**To verify the lineorder file is there**

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

**To verify the dwdate file is there**

hadoop fs -ls /user/ec2-user/dwdate.tbl

**To verify the supplier file is there**

hadoop fs -ls /user/ec2-user/supplier.tbl

**Go back into PIG**

bin/pig

**Load the lineorder File**

lineorderlarge = LOAD 'lineorder.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);

X = LIMIT lineorderlarge 10;

DUMP X;

**Load the dwdate File**

dwdatelarge = LOAD 'dwdate.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\_numinmonth:int, d\_daynuminyear:int, d\_weeknuminyear:int,

Y = LIMIT dwdatelarge 10;

DUMP Y;

d\_sellingseason:chararray, d\_lastdayinweekfl:chararray, d\_lastdayinmonthfl:chararray, d\_holidayfl:chararray, d\_weekdayfl:chararray);

**Load the supplier File**

supplierlarge = LOAD 'supplier.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);

Z = LIMIT supplierlarge 10;

DUMP Z;

**Create the Join, Verify it works by only outputting 10 columns**

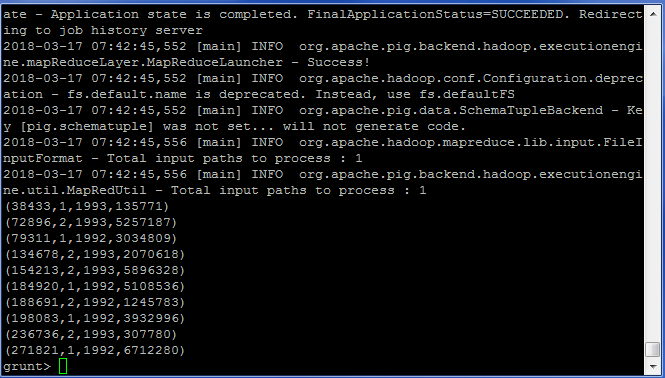
A = JOIN lineorderlarge BY lo\_orderdate, dwdatelarge BY d\_datekey;

B = JOIN A BY lo\_suppkey, supplierlarge BY s\_suppkey;

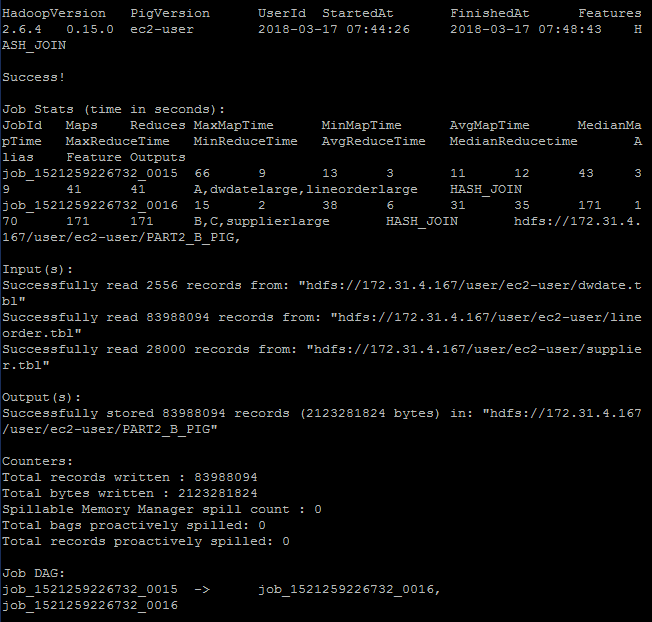
C = FOREACH B GENERATE lo\_partkey, lo\_suppkey, d\_year, lo\_revenue;

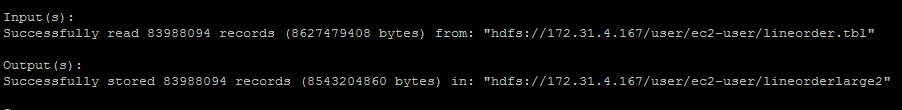
D = LIMIT C 10;

Describe D; Dump D;



Describe C; Store C INTO 'PART2\_B\_PIG' USING PigStorage ('\t');





**PART 2 B: PIG – Single Node Cluster**

Create a pre-join (i.e. a new data file) that corresponds to the following query below. You can think of it as a materialized view. What is the size of the new file? Use Hive and Pig (2 different solutions and be sure to report the file size for both).

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, lo\_supplier WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

INSERT OVERWRITE DIRECTORY 'hive\_pre-join\_file'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT lo\_partkey, lo\_suppkey, s\_suppkey, d\_year, lo\_revenue

FROM lineorder, dwdate, supplier

WHERE lo\_orderdate = d\_datekey and lo\_suppkey = s\_suppkey;

cd $PIG\_HOME

**To insure the -mkdir file exists**

hadoop fs -mkdir /user/ec2-user

**To place the lineorder file where PIG can retrieve it**

hadoop fs -put ../lineorder.tbl /user/ec2-user/

**To place the dwdate file where PIG can retrieve it**

hadoop fs -put ../dwdate.tbl /user/ec2-user/

**To place the supplier file where PIG can retrieve it**

hadoop fs -put ../supplier.tbl /user/ec2-user/

**To verify the lineorder file is there**

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

**To verify the dwdate file is there**

hadoop fs -ls /user/ec2-user/dwdate.tbl

**To verify the supplier file is there**

hadoop fs -ls /user/ec2-user/supplier.tbl

**Go back into PIG**

bin/pig

**Load the lineorder File**

Lineorderlarge\_single\_node = LOAD 'lineorder.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);

X = LIMIT Lineorderlarge\_single\_node 10;

DUMP X;

**Load the dwdate File**

Dwdatelarge\_single\_node = LOAD 'dwdate.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\_numinmonth:int, d\_daynuminyear:int, d\_weeknuminyear:int, d\_sellingseason:chararray, d\_lastdayinweekfl:chararray, d\_lastdayinmonthfl:chararray, d\_holidayfl:chararray, d\_weekdayfl:chararray);

Y = LIMIT Dwdatelarge\_single\_node 10;

DUMP Y;

**Load the supplier File**

Supplierlarge\_single\_node = LOAD 'supplier.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);

Z = LIMIT Supplierlarge\_single\_node 10;

DUMP Z;

**Create the Join, Verify it works by only outputting 10 columns**

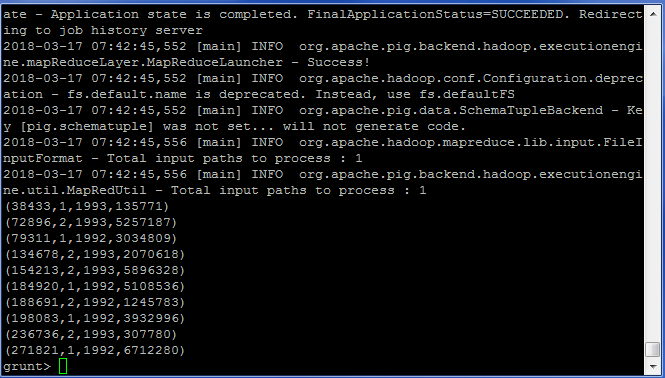
A = JOIN Lineorderlarge\_single\_node BY lo\_orderdate, Dwdatelarge\_single\_node BY d\_datekey;

B = JOIN A BY lo\_suppkey, Supplierlarge\_single\_node BY s\_suppkey;

C = FOREACH B GENERATE lo\_partkey, lo\_suppkey, d\_year, lo\_revenue;

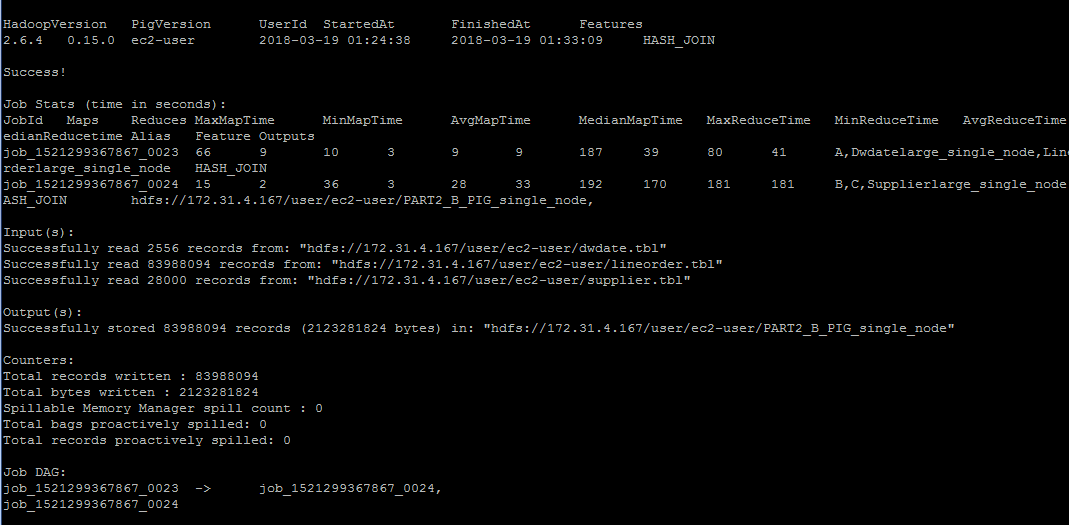
D = LIMIT C 10;

Describe D; Dump D;





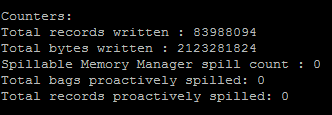
Describe C; Store C INTO 'PART2\_B\_PIG\_single\_node' USING PigStorage ('\t');



The time it took.



The size of the file



**Part 3: Clustering**

Using the file you have created in 1-B, run KMeans clustering using 11 clusters.

**A**. Using Mahout synthetic clustering as you have in a previous assignment on sample data. This entrails running the same clustering command, but substituting your own input data instead of the sample.

**NOTE:** if you get a java.lang.OutOfMemoryError error, you will need to reconfigure Hadoop to supply the java virtual machine with more memory. You can do this by editing the mapred-site.xml (Mapper should not need much RAM):

*<property>*

*<name> mapreduce.reduce.java.opts</name>*

*<value>-Xmx1024m</value>*

*</property>*

The amount of memory can be tweaked (you can go higher, but keep in mind how much physical memory your machine has). If you **still** run out of memory in 3-A submit the screenshot of that change and you will get full credit for the question.

To print working directory

Pwd

To move you up one directory

cd ..

In order to change directory to mahout use

cd /home/ec2-user/apache-mahout-distribution-0.11.2/

Change directory to where the files are located

cd /home/ec2-user

To Copy file performed in Part1 B to local

hadoop fs -copyToLocal Part\_1\_B/000001\_0

Insure the file is there and contains the data

nano 000001\_0

Create test directory in HDFS

hadoop fs -mkdir -p testdata

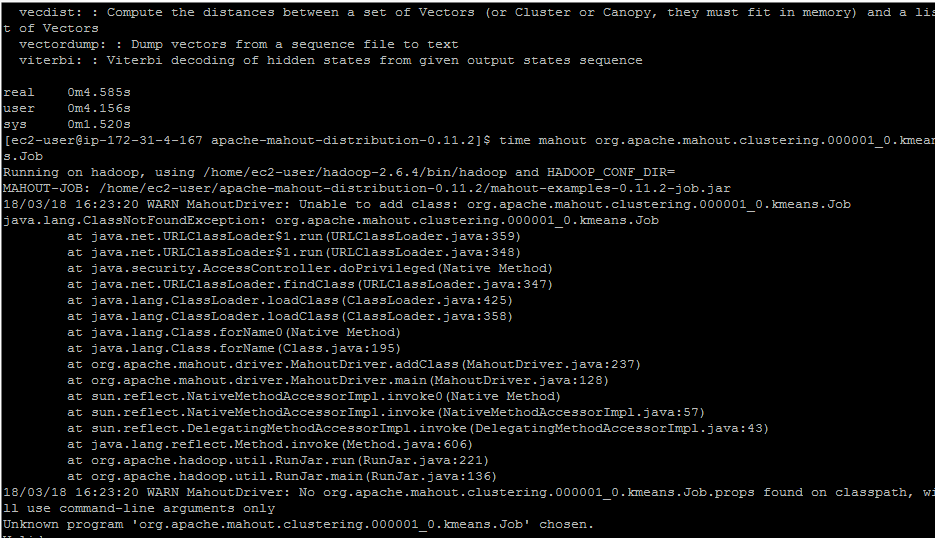
Inspect one of the files created in Part1 B from above.

hadoop fs -cat Part\_1\_B/000001\_0

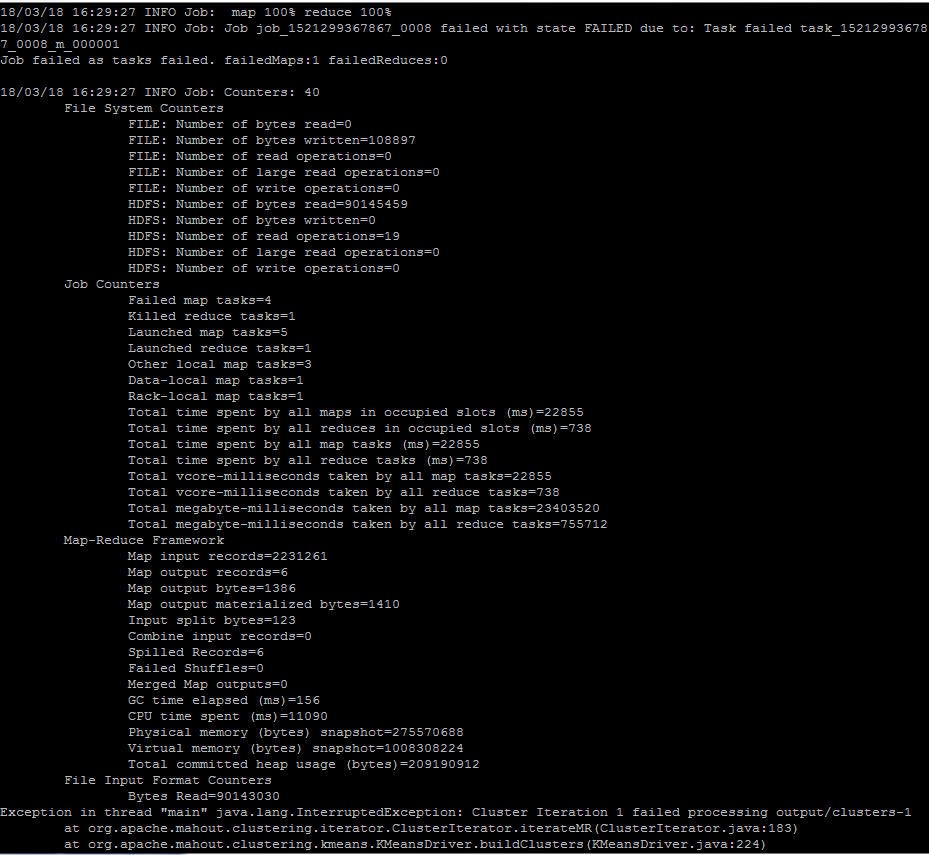
Copy the file into testdata

hadoop fs -put 000001\_0 testdata/

time mahout org.apache.mahout.clustering.000001\_0.kmeans.Job



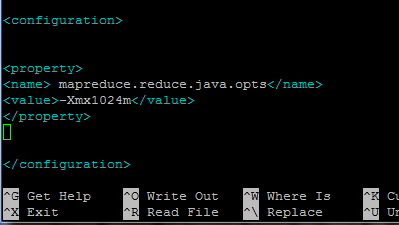
I was unable to get the cluster to operate correctly as it kept coming up with the statement “Unable to add class..” “Unknown program chose…” I then went back to Assignment 4 and tried to re-run that exercise (which I ran successfully before); but still had issues. (See next screen shot) It appeared to start out alright but then came with the error… “Job failed as tasks failed. failedMaps:1 failedReduces:0..”



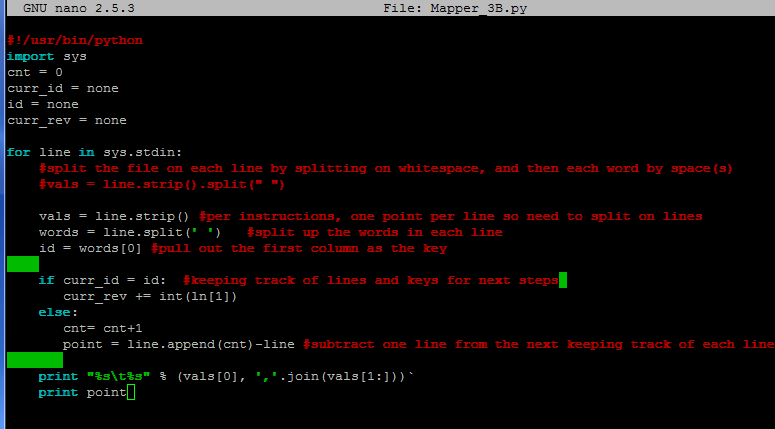
(clusterdump is a built-in Mahout command that will produce the result of KMeans. Output file is written to clusters-10-final because that is where the output is written after 10 iterations. The center points are placed in a separate file, called clusteredPoints)

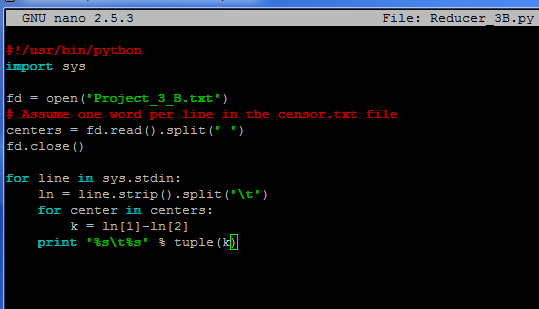
mahout clusterdump --input output/clusters-11-final --pointsDir output/clusteredPoints –output clusteranalyze1.txt

The file clusteranalyze1.txt should contains the results of the Kmeans run after 11 iterations



**B.** Using Hadoop streaming perform three iterations manually (initially with randomly chosen centers). This would require passing a text file with cluster centers using -file option, opening the centers.txt in the mapper with open(‘centers.txt’, ‘r’) and assigning a key to each point based on which center is the closest to each particular point. Your reducer would then compute the new centers, and at that point the iteration is done and the output of the reducer can be given to the next pass.





Ideally, the following code would start the Hadoop streaming process. Taking in the file(s) from Part1\_B, and outputting them to data/output. The mapper passes the files with the cluster centers to the reducer. The reducer would then perform the calculations to determine the clusters surrounding the center point.

hadoop jar hadoop-streaming-2.6.4.jar -input /data/000001\_0 -output /data/output -mapper mapper\_3B.py -reducer Reducer\_3B.py -file mapper\_3B.py -file Reducer\_3B.py -file Project\_3\_B

**Part 4: Performance**

Compare the performance given following combinations.

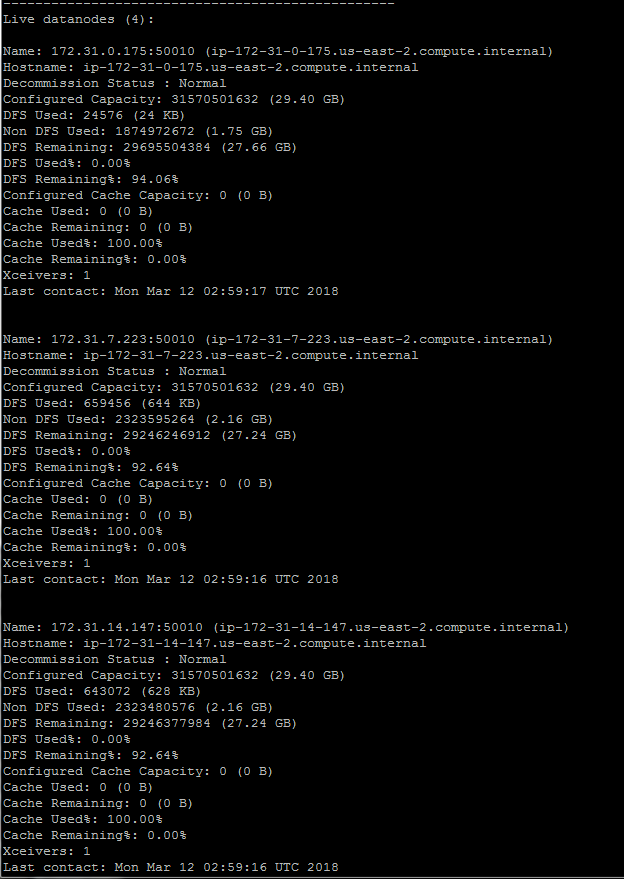
**A. All three of your solutions to Part-1A with**

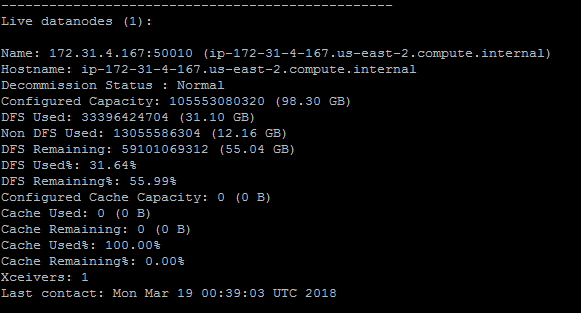
**a. Scale4: a single node cluster and a cluster of at least 4 nodes**

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

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

**C. Summarize the results and cluster performance/scaling in at least a paragraph.**





|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | HIVE | | PIG | | Hadoop-Streaming | |
|  | 5-Node | Single-Node | 5-Node | Single-Node | 5-Node | Single-Node |
| Part1-A | 2:05 | 2:07 | 1:07 | 4:48 | Couldn’t get working ☹ | Couldn’t get working ☹ |
| Part2-B | :39 | 1:54 | 4:17 | 9:47 | Couldn’t get working ☹ | Couldn’t get working ☹ |

It was interesting to see the differences between a 5-node cluster and a single node cluster for the different techniques.

For Hive, there wasn’t an extremely noticeable difference between a 5-node cluster and a single node cluster, with only a couple minutes difference when compiling a more complex query, which tells me that when performing a complex query, it is best to use Hive to save time and clusters. When converting an entire file to comma deliminted, there were only a couple seconds difference (I ran this twice to be confirm). This is also very telling as to even less complex queries show limited deltas in time. Hive is the way to go for me as its faster and also more familiar code to write (SQL).

For PIG, the deltas in performance were extremely noticeable between a 5-node cluster and single-node cluster. It was close to 5x longer when converting the file to comma delimited, which makes sense as that’s how many more clusters were being used. However, it was painful to watch the queries run as I was already conditioned with the Hive cluster. If you are forced to use PIG, the more clusters the better when it comes to performance. However, PIG is a new form of code and I would prefer Hive because I know the code better and also it performs better.

**Extra Credit**

Research and describe the most affordable way to build a 10-Petabyte drive. The drive should be built to own, not to rent (Dropbox or similar services doesn’t count, even if it does say “unlimited” storage).

Submit a single document containing your written answers. Be sure that this document contains your name and “CSC 555 Project Phase 2” at the top.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The most affordable way to build a 10-Petabyte drive to own is by using many of the key fundamentals of Backblaze’s open-source hardware design.  When stripped away of everything but the core function of what all the big enterprise cloud brands do, what you get is as simple as transferring data to and from a hard drive over the internet. Below is an outline of how you would build out this drive from scratch:

1. **What you need to buy:**

To build out your own cloud server, you must first remember that the hard drives purchased will largely determine the price point and make up the bulk–estimate at least half and as much as 80 percent–of the total investment. The software is free, so the remainder comes from the enclosure, racks, and all the components. The open-source design for Backblaze’s newest enclosure claims it can bring the cost down to as little as 3.6 cents per GB of storage, totaling 240TB with all 60 drives accounted for. The pods sit in 4U racks and the 6.0 design extends a few inches past the edge, so consider how much space you have in your server room.

* You can buy one of these “pods” pre-built for between $3,000 and $7,000, depending on how many hard drive slots you need, from [45 Drives](http://www.45drives.com/) or [BackupPods](https://www.backuppods.com/" \t "_blank). With the pre-built pod, the only other thing you need to buy is the actual hard drives.
* Or you can build one yourself. For a 6th-gen pod with 60 hard drives, the full parts lists along with estimated prices can be found in [this PDF](https://f001.backblazeb2.com/file/Backblaze_Blog/Storage-Pod-6/Storage+Pod+6.0+Parts+List.pdf). Most components can be found on Newegg and Amazon, but some will come from special distributors or contract assemblers.

While the parts vary for each version of Backblaze’s design, here’s a generic rundown of everything you’ll need:

* 4U chassis
* Power supply
* On/Off switch
* Case fan
* Dampeners
* Fan mounts
* Motherboard
* CPU Fan
* CPU (Intel)
* 8GB RAM
* Port multiplier backbplanes
* SATA III cards
* SATA III cables
* Cable harnesses
* Screws and cable ties

**2. Once you have all the parts, it’s time to start assembly. You can download the following from BackBlaze:**

* [Wiring diagrams](https://f001.backblaze.com/file/Backblaze_Blog/Storage-Pod-6/Wiring+Diagrams.zip) (ZIP file)
* [Wiring routes](https://f001.backblaze.com/file/Backblaze_Blog/Storage-Pod-6/Wiring+Routes.zip) (ZIP file)
* [Build book](https://f001.backblaze.com/file/Backblaze_Blog/Storage-Pod-6/Build+Book+Backblaze+60+Drive.pdf) (PDF)

All in all the 60-drive set up costs an estimated $3,500 according to BackBlaze. Remember that doesn’t include the drives. With 45 4TB hard drives, the total bill comes out to about $10,500.

**3. Creating a cloud:**

Once you’ve built a huge storage server for a fraction of what it would have cost you to use someone else’s servers, you still need to make it into a cloud that’s accessible to clients, staff, and/or applications.

Let’s work from the bottom up. Backblaze recommends 64-bit Debian Linux as the operating system. The fdisk tool is used to create one partition per drive. If you bought one of the pre-made pods, many of the drivers will come pre-installed. Drives are clustered in sets of 15 into RAID6 volumes with two parity drives each using the mdadm utility.

Now you must choose between the JFS or ext4 filesystem. Ext4 is more common but JFS is what BackBlaze uses. Each pod has its own HTTPS IP address, which is how it will be accessed. ext4 supports up to 1EiB, but the distro copy of e2fsprogs only supported 16TB. Building from source using the 64bit flag solves this.

Once all that is in place, you end up with about 83 percent of usable space out of the total. This is where you stop relying on Backblaze for advice, as its cloud software is proprietary.

You have a couple options for cloud software. NFS is tried and tested on Linux but not all that compatible with mobile devices.

Another option is [Oxygen Cloud](https://home.oxygencloud.com/), which [uses the Oxygen Storage Connector](http://zuhaiblog.com/2012/03/30/how-to-make-your-backblaze-pod-be-powered-by-the-cloud/) to convert used storage on the server into storage that can be used with Oxygen Cloud apps. Oxygen cloud encrypts data in transit end-to-end. You also get access to Oxygen Tunnel Gateways, which allow you to access your storage from outside your own firewall without having to change your configuration.

**4. Drawbacks:**

Before you go to Newegg and start filling up your shopping cart with hard drives and components, it’s important to consider the potential drawbacks of not going with a provider like Amazon S3 or EMC.

The biggest risk is that you could lose data. That means you could lose your job and/or harm your company and coworkers. The system uses a single disk for the host operating system, some do not have redundant or failover power supplies, and any health or monitoring software must be built, installed, and configured by hand.

It’s also not as easy to expand or maintain. When you pay for Amazon S3, maintenance is all taken care of for you. But a task as simple as swapping out a failed drive in a DIY storage pod requires you remove 12 screws and the top cover, not to mention setting up custom wiring harnesses. Much of the necessary maintenance will require the system to be taken offline and possibly powered down.

The simplest solution to both of these problems is to make an extra one or more redundant servers that act as failovers.

***Personally,*** based on all the research I did, it makes more sense to rent back up storage from Backblaze (see article outlined below) as they have full parts list and diagrams showing how they put everything together. For the purpose of this extra credit assignment, we were to focus on how we would build from scratch and OWN which is what is outlined above.

<https://secure.backblaze.com/petabytes-on-a-budget-how-to-build-cheap-cloud-storage.pdf>