CSC 555: Mining Big Data

Project, Phase 1 (due Sunday, October 22nd)

In this part of the project, you will 1) Set up a 3-node cluster and 2) perform data warehousing and transformation queries using Hive, Pig and Hadoop streaming. The modified Hive-style schema is still available at:
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql
It is based on SSBM benchmark (derived from industry standard TPCH benchmark). I modified it from SQL to HiveQL. This is Scale1, or the smallest unit – lineorder is the largest table at about 0.6GB. You can use wget to download the following links. Keep in mind that data is |-separated (not csv).
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/dwdate.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl
Please be sure to submit all code (pig and python).

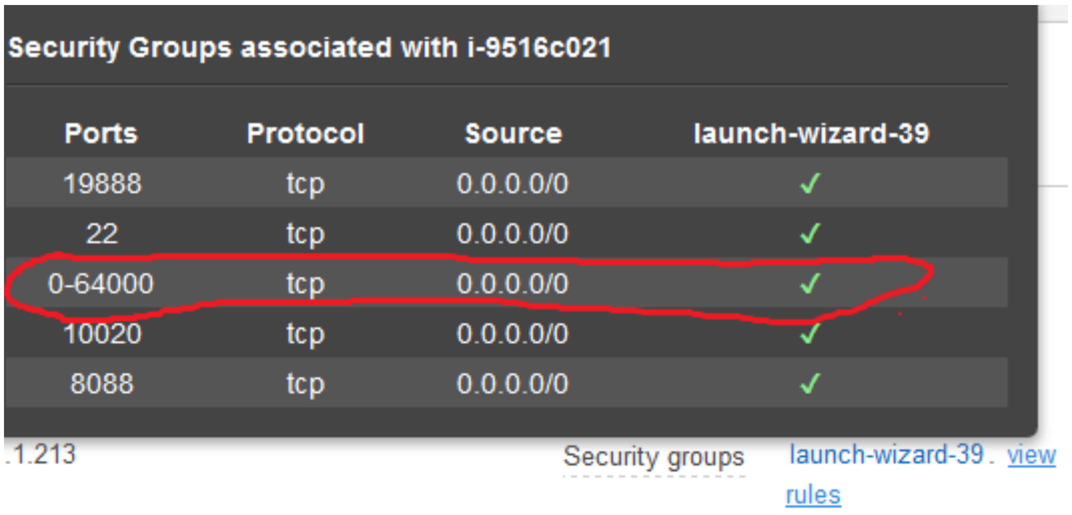
Part 1: Multi-node cluster

1) Your first step is to setup a multi-node cluster and re-run a simple wordcount. For this part, you will create a 3-node cluster (with a total of 1 master + 2 worker nodes). Include your master node in the "slaves" file, to make sure all 3 nodes are working.
You need to perform the following steps:

   1. Create a new node of a medium size. It is possible, but I do not recommend trying to reconfigure your existing Hadoop into this new cluster (it is much easier to make 3 new nodes for a total of 4 in your AWS account).

      a. When creating a node I recommend changing the default 8G hard drive to 30G so that you do not run out of space easily.

      b. Change your security group setting to open firewall access. Rather than figure out all individual port, you can set 0-64000 range opening up all ports (not the most secure setting in the long term, but fine for us)



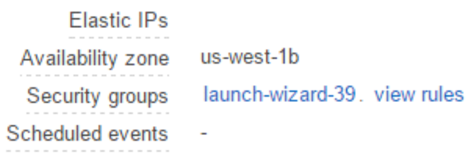
Security Groups associated with i-9516c021

Ports	Protocol	Source	launch-wizard-39
19888	tcp	0.0.0.0/0	✓
22	tcp	0.0.0.0/0	✓
0-64000	tcp	0.0.0.0/0	✓
10020	tcp	0.0.0.0/0	✓
8088	tcp	0.0.0.0/0	✓

.1.213 Security groups launch-wizard-39. view rules

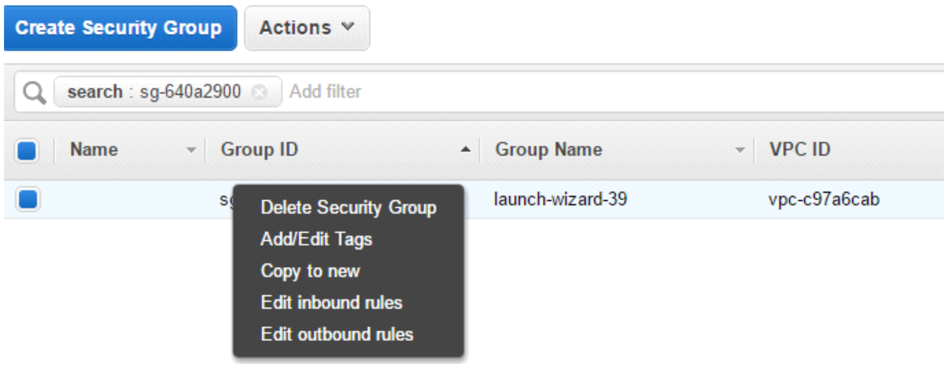
c. Step by step instructions on how to make the change to open up the ports:

Click on security group (launch-wizard-x)



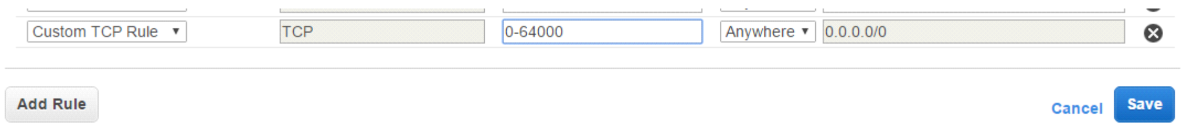
Elastic IPs	
Availability zone	us-west-1b
Security groups	launch-wizard-39. view rules
Scheduled events	-

Right click on the security group and choose Edit inbound rules



Name	Group ID	Group Name	VPC ID
	sg	launch-wizard-39	vpc-c97a6cab

Add a new rule and put in the ports 0-64000 and "Anywhere" and click save.



Custom TCP Rule | TCP | 0-64000 | Anywhere | 0.0.0.0/0

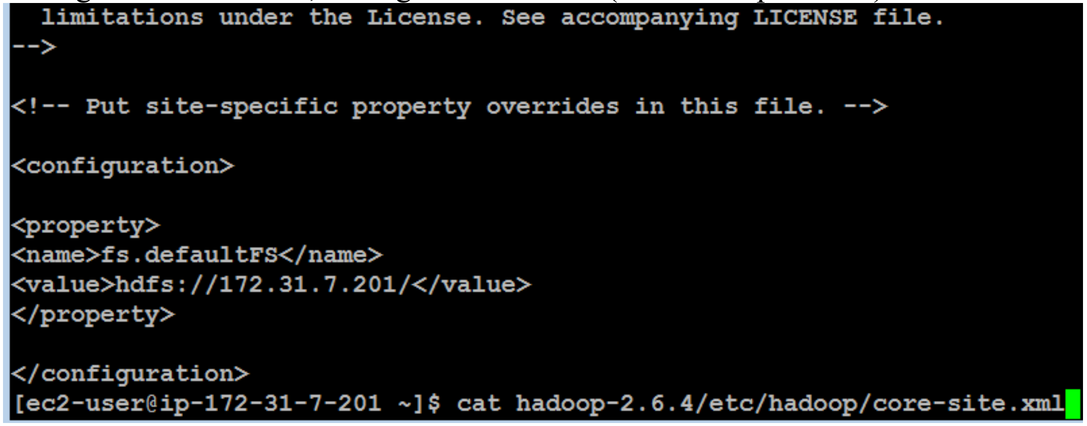
This will open the firewall completely for all ports.

d. Finally, right click on the Master node and choose "create more like this" to create 2 more nodes with same settings. If you configure the network settings on master first, security group information will be copied.

NOTE: Hard drive size will not be copied and default to 8G unless you change it.

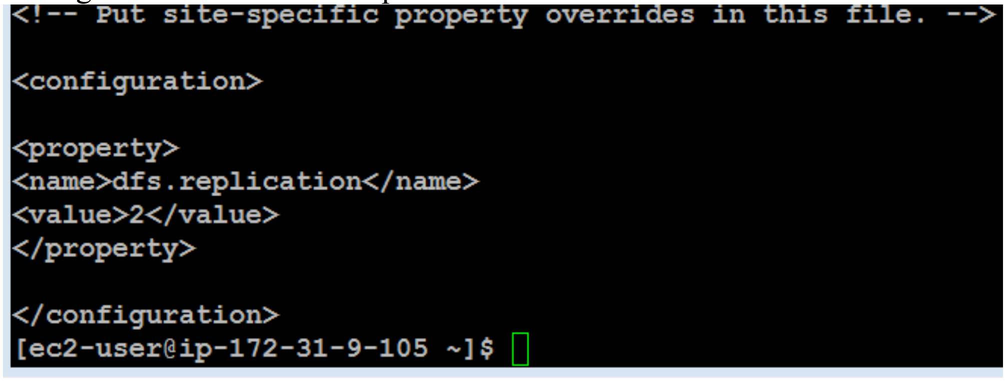
2. Connect to the master and set up Hadoop similarly to what you did previously. Do not attempt to repeat these steps on workers yet – you will only need to set up Hadoop once.

   a. Configure core-site.xml, adding the **PrivateIP** (do not use public IP) of the master.



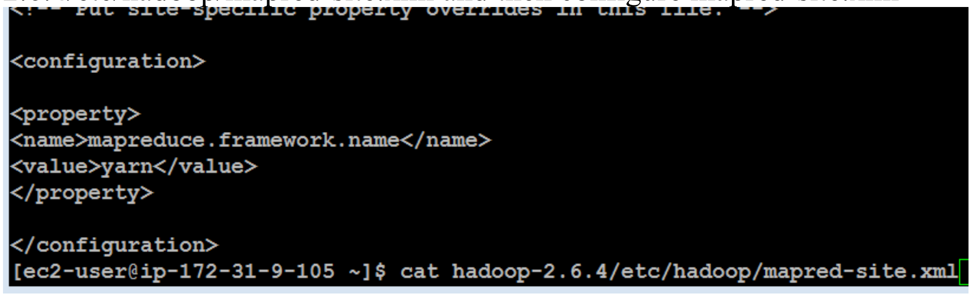
```
 limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://172.31.7.201/</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/core-site.xml
```

   b. Configure hdfs-site and set replication factor to 2.



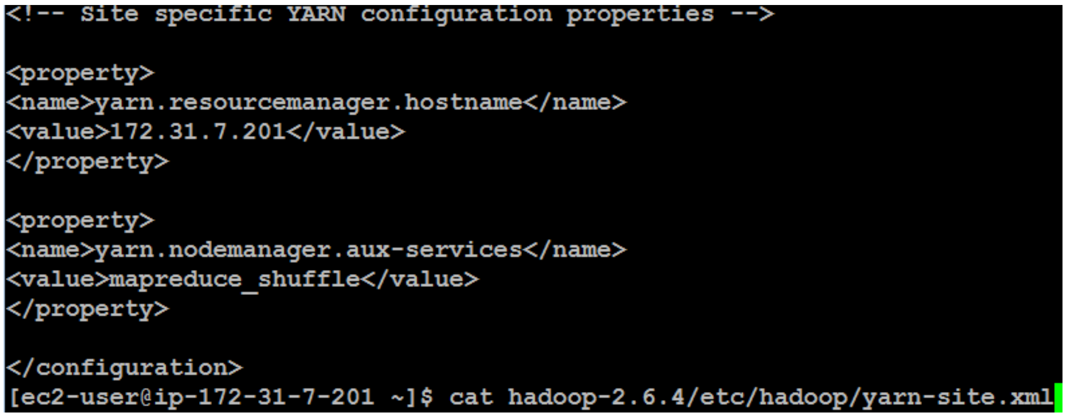
```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>dfs.replication</name>
<value>2</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$
```

   c. cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml and then configure mapred-site.xml



```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$ cat hadoop-2.6.4/etc/hadoop/mapred-site.xml
```

   d. Configure yarn-site.xml (once again, use PrivateIP of the master)



```
<!-- Site specific YARN configuration properties -->

<property>
<name>yarn.resourcemanager.hostname</name>
<value>172.31.7.201</value>
</property>

<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/yarn-site.xml
```

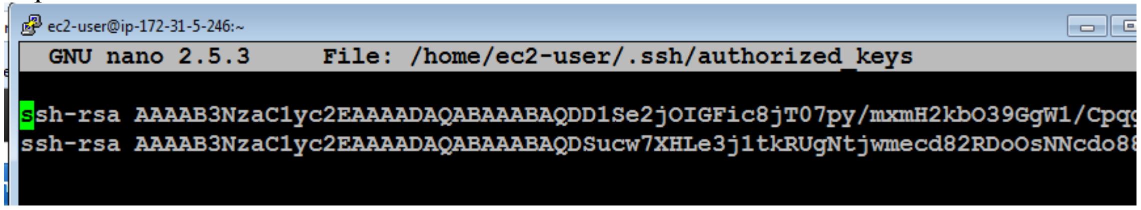
Finally, edit the slaves file and list your 3 nodes (master and 2 workers) using Private IPs

```
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/slaves
172.31.7.201
172.31.5.246
172.31.11.50
```

Make sure that you use private IP (private DNS is also ok) for your configuration files (such as conf/masters and conf/slaves or the other 3 config files). The advantage of the Private IP is that it does not change after your instance is stopped (if you use the Public IP, the cluster would need to be reconfigured every time it is stopped). The downside of the Private IP is that it is only meaningful within the Amazon EC2 network. So all nodes in EC2 can talk to each other using Private IP, but you cannot connect to your instance from the outside (e.g., from your laptop) because Private IP has no meaning for your laptop (since your laptop is not part of the Amazon EC2 network).

Now, we will pack up and move Hadoop to the workers. All you need to do is to generate and then copy the public key to the worker nodes to achieve passwordless access across your cluster.

1. Run ssh-keygen -t rsa (and enter empty values for the passphrase) on the master node. That will generate .ssh/id_rsa and .ssh/id_rsa.pub (private and public key). You now need to manually copy the .ssh/id_rsa.pub and append it to ~/.ssh/authorized_keys **on each worker.**
   Keep in mind that this is a single-line public key and accidentally introducing a line break would break it. For example:
   Note that this is NOT the master, but one of the workers (ip-172-31-5-246). The first public key is the .pem Amazon half and the 2nd public key is the master's public key copied in as one line.

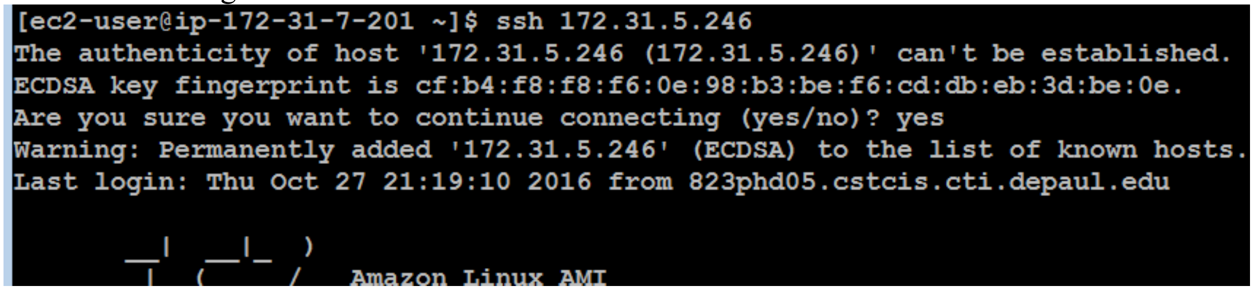


```
ec2-user@ip-172-31-5-246:~
  GNU nano 2.5.3      File: /home/ec2-user/.ssh/authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDD1Se2jOIGFic8jT07py/mxmH2kbO39GgW1/Cpqg
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSucw7XHLe3j1tkRUgNtjwmecd82RDoOsNNcdo88
```

You can add the public key of the master to the master by running this command:

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

Make sure that you can ssh to all of the nodes from the master node (by running ssh 54.186.221.92, where the IP address is your worker node) from the master and ensuring that you were able to login. You can exit after successful ssh connection by typing exit (the command prompt will tell you which machine you are connected to, e.g., ec2-user@ip-172-31-37-113). Here's me ssh-ing from master to worker.



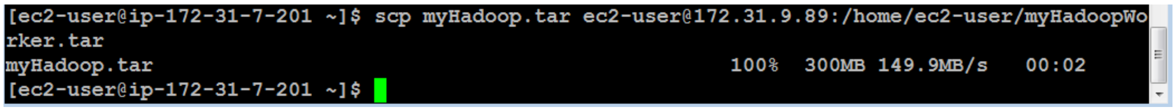
```
[ec2-user@ip-172-31-7-201 ~]$ ssh 172.31.5.246
The authenticity of host '172.31.5.246 (172.31.5.246)' can't be established.
ECDSA key fingerprint is cf:b4:f8:f8:f6:0e:98:b3:be:f6:cd:db:eb:3d:be:0e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.5.246' (ECDSA) to the list of known hosts.
Last login: Thu Oct 27 21:19:10 2016 from 823phd05.cstcis.cti.depaul.edu

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
```

Once you are have verified that you can ssh from the master node to every cluster member including the master itself (ssh localhost), you are going to return to the master node (exit until your prompt shows the IP address of the master node) and pack up the contents of the hadoop directory there. Make sure your Hadoop installation is configured correctly (because from now on, you will have 3 copies of the Hadoop directory and all changes may need to be applied in 3 places).

cd (go to root home directory, i.e. /home/ec2-user/)
(pack up the entire Hadoop directory into a single file for transfer. You can optionally compress the file with gzip)
tar cvf myHadoop.tar hadoop-2.6.4
ls -al myHadoop.tar (to verify that the .tar file had been created)

Now, you need to copy the myHadoop.tar file to every non-master node in the cluster. If you had successfully setup public-private key access in the previous step, this command (for each worker node) will do that:

(copies the myHadoop.tar file from the current node to a remote node into a file called myHadoopWorker.tar. Don't forget to replace the IP address with that your worker nodes. By the way, since you are on the Amazon EC2 network, either Public or Private IP will work just fine.)
scp myHadoop.tar ec2-user@54.187.63.189:/home/ec2-user/myHadoopWorker.tar



```
[ec2-user@ip-172-31-7-201 ~]$ scp myHadoop.tar ec2-user@172.31.9.89:/home/ec2-user/myHadoopWo
rker.tar
myHadoop.tar                                      100%  300MB 149.9MB/s   00:02
[ec2-user@ip-172-31-7-201 ~]$
```

Once the tar file containing your Hadoop installation from master node has been copied to each worker node, you need to login to each non-master node and unpack the .tar file.

Run the following command (on each worker node, not on the master) to untar the hadoop file. We are purposely using a different tar archive name (i.e., myHadoopWorker.tar), so if you get "file not found" error, that means you are running this command on the master node or have not successfully copied myHadoopWorker.tar file.

```
tar xvf myHadoopWorker.tar
```

Once you are done, run this on the master (nothing needs to be done on the workers to format the cluster unless you are re-formatting, in which case you'll need to delete the dfs directory).

```
hadoop namenode -format
```

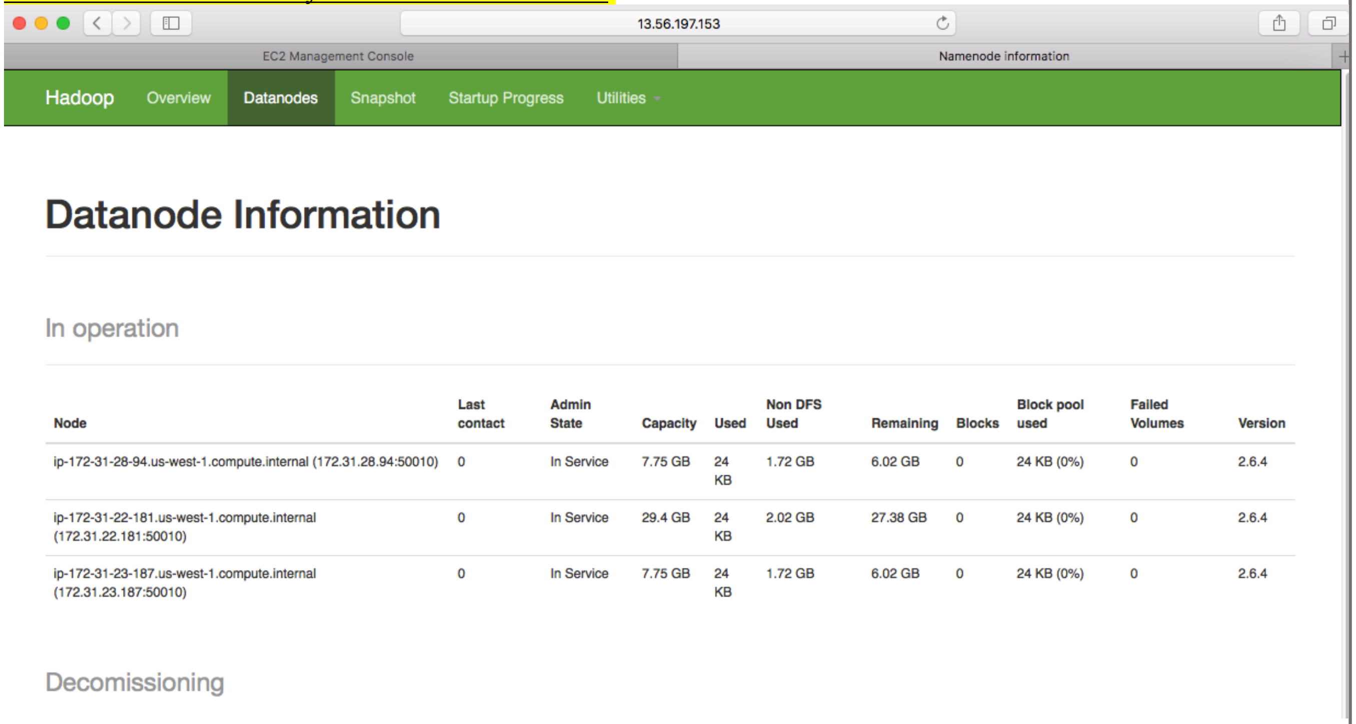
Once you have successfully completed the previous steps, you should be able to start and use your new cluster by going to the master node and running the start-dfs.sh and start-yarn.sh scripts (you do not need to explicitly start anything on worker nodes – the master will do that for you).

You should verify that the cluster is running by pointing your browser to the link below.

http://[insert-the-public-ip-of-master]:50070/

Make sure that the cluster is operational (you can see the 3 nodes under Datanodes tab).

Submit a screenshot of your cluster status view.



Datanode Information

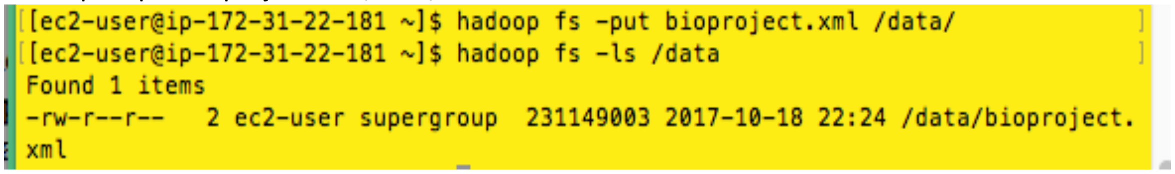
In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-28-94.us-west-1.compute.internal (172.31.28.94:50010)	0	In Service	7.75 GB	24 KB	1.72 GB	6.02 GB	0	24 KB (0%)	0	2.6.4
ip-172-31-22-181.us-west-1.compute.internal (172.31.22.181:50010)	0	In Service	29.4 GB	24 KB	2.02 GB	27.38 GB	0	24 KB (0%)	0	2.6.4
ip-172-31-23-187.us-west-1.compute.internal (172.31.23.187:50010)	0	In Service	7.75 GB	24 KB	1.72 GB	6.02 GB	0	24 KB (0%)	0	2.6.4

Decomissioning

Repeat the steps for wordcount using bioproject.xml from Assignment 1 and submit screenshots of running it.

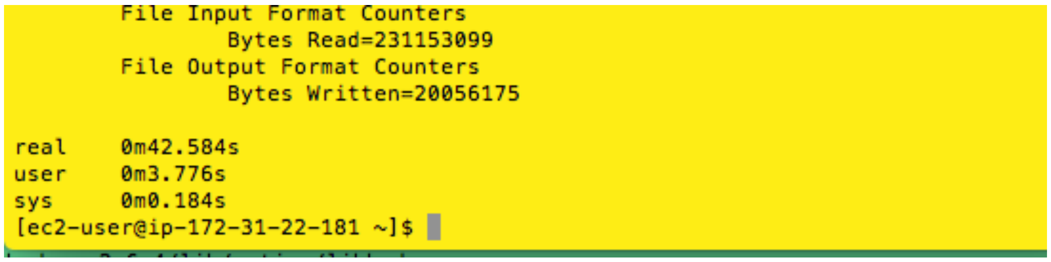
hadoop fs -put bioproject.xml /data/



```
[ec2-user@ip-172-31-22-181 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-22-181 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   2 ec2-user supergroup  231149003 2017-10-18 22:24 /data/bioproject.
xml
```

Report the time:

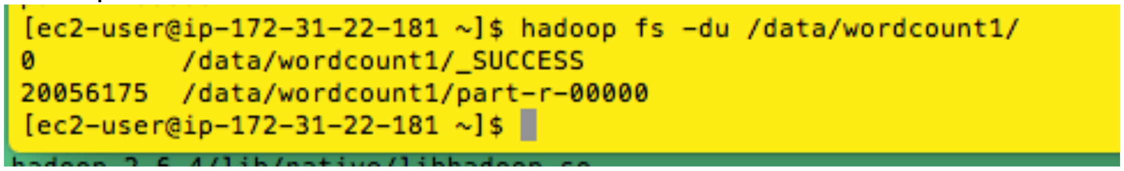
time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar wordcount /data/bioproject.xml /data/wordcount1



```
        File Input Format Counters
                Bytes Read=231153099
        File Output Format Counters
                Bytes Written=20056175

real    0m42.584s
user    0m3.776s
sys     0m0.184s
[ec2-user@ip-172-31-22-181 ~]$
```

Report the size of the wordcount file

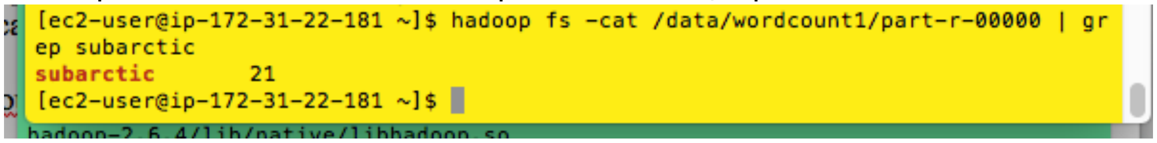
hadoop fs -du /data/wordcount1/



```
[ec2-user@ip-172-31-22-181 ~]$ hadoop fs -du /data/wordcount1/
0         /data/wordcount1/_SUCCESS
20056175  /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-22-181 ~]$
```

Report the count of grep the word "subarctic"

hadoop fs -cat /data/wordcount1/part-r-00000 | grep subarctic



```
[ec2-user@ip-172-31-22-181 ~]$ hadoop fs -cat /data/wordcount1/part-r-00000 | gr
ep subarctic
subarctic       21
[ec2-user@ip-172-31-22-181 ~]$
```

Part 2: Hive

Run the following five (1.1, 1.2, 1.3 and 2.1, 2.2) queries in Hive and record the time they take to execute:
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_queries.sql

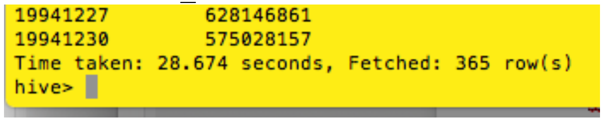
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/dwdate.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl

```
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 '|';


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

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 '|';

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

```
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 '|';

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 '|';

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

1.1)

```
select lo_orderdate, sum(lo_extendedprice) as revenue
from lineorder, dwdate
where lo_orderdate = d_datekey
  and d_year = 1994
  and lo_discount between 1 and 3
  and lo_quantity < 25
GROUP BY lo_orderdate;
```

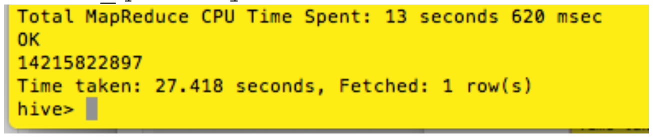


```
19941227        628146861
19941230        575028157
Time taken: 28.674 seconds, Fetched: 365 row(s)
hive>
```

Ans: 28.674 sec.

1.2)

```
select sum(lo_extendedprice) as revenue
from lineorder, dwdate
where lo_orderdate = d_datekey
and d_yearmonth = 'Jan1993'
and lo_discount between 5 and 6
and lo_quantity between 25 and 35;
```

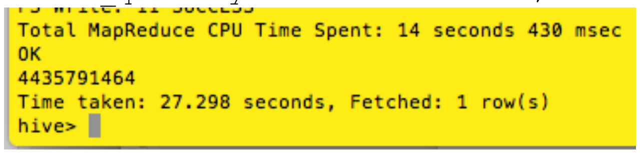


```
Total MapReduce CPU Time Spent: 13 seconds 620 msec
OK
14215822897
Time taken: 27.418 seconds, Fetched: 1 row(s)
hive>
```

Ans: 27.418 sec.

1.3)

```
select sum(lo_extendedprice) as revenue
from lineorder, dwdate
where lo_orderdate = d_datekey
and d_weeknuminyear = 6 and d_year = 1994
and lo_discount between 5 and 8
and lo_quantity between 36 and 41;
```

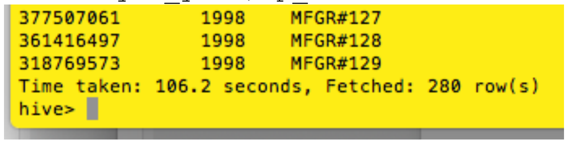


```
Total MapReduce CPU Time Spent: 14 seconds 430 msec
OK
4435791464
Time taken: 27.298 seconds, Fetched: 1 row(s)
hive>
```

Ans: 27.298 sec.

2.1)

```
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;
```

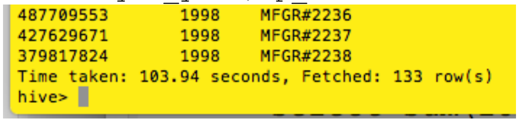


```
377507061       1998    MFGR#127
361416497       1998    MFGR#128
318769573       1998    MFGR#129
Time taken: 106.2 seconds, Fetched: 280 row(s)
hive>
```

Ans: 106.2 sec.

2.2)

```
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_brand1 between 'MFGR#2221'
and 'MFGR#2238'
and s_region = 'ASIA'
group by d_year, p_brand1
order by d_year, p_brand1;
```



```
487709553       1998    MFGR#2236
427629671       1998    MFGR#2237
379817824       1998    MFGR#2238
Time taken: 103.94 seconds, Fetched: 133 row(s)
hive>
```

Ans: 103.94 sec.

Part 3: Pig

Convert and load the data into Pig, implementing queries 0.1, 0.2, 0.3. You only need to do all of the queries for Pig. Check disk storage, if your disk usage is over 90% Pig may hang without an error or a warning.

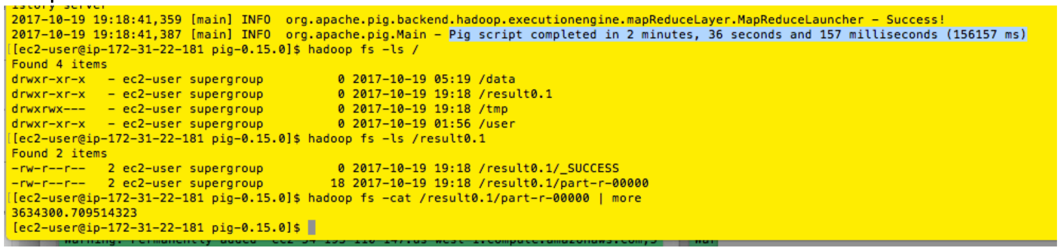
One easy way to time Pig is as follows: put your sequence of pig commands into a text file and then run, from command line in pig directory (e.g., [ec2-user@ip-172-31-6-39 pig-0.15.0]$), bin/pig -f pig_script.pig (which will inform you how long the pig script took to run).

```
--Q0.1
SELECT AVG(lo_revenue)
FROM lineorder;
```

<u>Parse the followings on the script and then execute in the PIG directory</u>

```
lod = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS (lo_orderkey :float,
  lo_linenumber       :float,
  lo_custkey          :float,
  lo_partkey          :float,
  lo_suppkey          :float,
  lo_orderdate        :float,
  lo_orderpriority    :chararray,
  lo_shippriority     : chararray,
  lo_quantity         :float,
lo_extendedprice    :float,
lo_ordertotalprice  :float,
lo_discount         :float,
lo_revenue          :float,
lo_supplycost       :float,
lo_tax              :float,
lo_commitdate       :float,
 lo_shipmode        : chararray);
by_all= group lod ALL;
all_avg = FOREACH by_all GENERATE AVG( lod.lo_revenue);
Dump all_avg;
STORE all_avg INTO '/result0.1' USING PigStorage ('\t');
```

Output:



```
2017-10-19 19:18:41,359 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-19 19:18:41,387 [main] INFO  org.apache.pig.Main - Pig script completed in 2 minutes, 36 seconds and 157 milliseconds (156157 ms)
[ec2-user@ip-172-31-22-181 pig-0.15.0]$ hadoop fs -ls /
Found 4 items
drwxr-xr-x   - ec2-user supergroup          0 2017-10-19 05:19 /data
drwxr-xr-x   - ec2-user supergroup          0 2017-10-19 19:18 /result0.1
drwxrwx---   - ec2-user supergroup          0 2017-10-19 19:18 /tmp
drwxr-xr-x   - ec2-user supergroup          0 2017-10-19 01:56 /user
[ec2-user@ip-172-31-22-181 pig-0.15.0]$ hadoop fs -ls /result0.1
Found 2 items
-rw-r--r--   2 ec2-user supergroup          0 2017-10-19 19:18 /result0.1/_SUCCESS
-rw-r--r--   2 ec2-user supergroup         18 2017-10-19 19:18 /result0.1/part-r-00000
[ec2-user@ip-172-31-22-181 pig-0.15.0]$ hadoop fs -cat /result0.1/part-r-00000 | more
3634300.709514323
[ec2-user@ip-172-31-22-181 pig-0.15.0]$
```

<u>Ans: 2min 36sec 157 ms was taken</u>

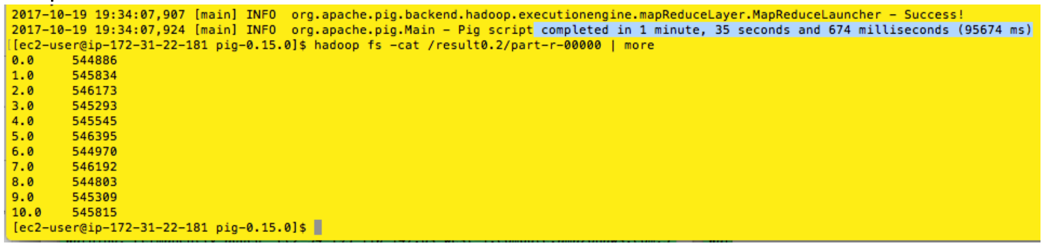
<u>--Q0.2</u>

```
SELECT lo_discount, COUNT(lo_extendedprice)
FROM lineorder
GROUP BY lo_discount;
```

<u>Parse the followings on the script and then execute in the PIG directory</u>

```
lod = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS (lo_orderkey :float,
  lo_linenumber       :float,
  lo_custkey          :float,
  lo_partkey          :float,
  lo_suppkey          :float,
  lo_orderdate        :float,
  lo_orderpriority     :chararray,
  lo_shippriority      : chararray,
  lo_quantity          :float,
lo_extendedprice     :float,
lo_ordertotalprice   :float,
lo_discount          :float,
lo_revenue           :float,
lo_supplycost        :float,
lo_tax               :float,
lo_commitdate        :float,
 lo_shipmode          : chararray);

by_discount = group lod by lo_discount;
discount_count = FOREACH by_discount GENERATE group as lo_discount, COUNT(lod);
dump discount_count;
store discount_count into '/result0.2' using PigStorage('\t');
```

Output:



```
2017-10-19 19:34:07,907 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-19 19:34:07,924 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute, 35 seconds and 674 milliseconds (95674 ms)
[ec2-user@ip-172-31-22-181 pig-0.15.0]$ hadoop fs -cat /result0.2/part-r-00000 | more
0.0     544886
1.0     545834
2.0     546173
3.0     545293
4.0     545545
5.0     546395
6.0     544970
7.0     546192
8.0     544803
9.0     545309
10.0    545815
[ec2-user@ip-172-31-22-181 pig-0.15.0]$
```

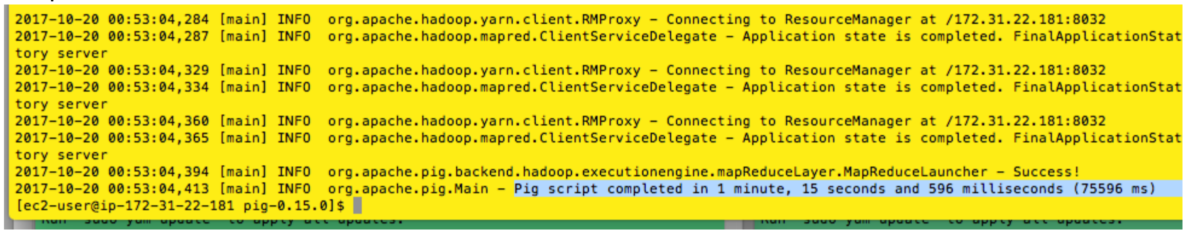
<u>Ans: 1min 35sec 674ms</u>

```
--Q0.3 Added simple test query
SELECT lo_quantity, SUM(lo_revenue)
FROM lineorder
WHERE lo_discount < 3
GROUP BY lo_quantity;
```

<u>Parse the followings on the script and then execute in the PIG directory</u>

```
lod = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS (lo_orderkey :float,
  lo_linenumber      :float,
  lo_custkey         :float,
  lo_partkey         :float,
  lo_suppkey         :float,
  lo_orderdate       :float,
  lo_orderpriority    :chararray,
  lo_shippriority     : chararray,
  lo_quantity        : chararray,
lo_extendedprice    :float,
lo_ordertotalprice  :float,
lo_discount         :float,
lo_revenue          :float,
lo_supplycost       :float,
lo_tax              :float,
lo_commitdate       :float,
 lo_shipmode        : chararray);
LowDis = filter lod by lo_discount < 3;
lod = foreach lod generate lo_quantity as qua, lo_revenue as rev;
by_lo_quantity = group lod by qua;
sumTab = FOREACH by_lo_quantity GENERATE group as qua, SUM(lod.rev);
dump sumTab;
store sumTab into '/result0.3' using PigStorage('\t');
```

Output:



```
2017-10-20 00:53:04,284 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /172.31.22.181:8032
2017-10-20 00:53:04,287 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStat
tory server
2017-10-20 00:53:04,329 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /172.31.22.181:8032
2017-10-20 00:53:04,334 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStat
tory server
2017-10-20 00:53:04,360 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /172.31.22.181:8032
2017-10-20 00:53:04,365 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStat
tory server
2017-10-20 00:53:04,394 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-10-20 00:53:04,413 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute, 15 seconds and 596 milliseconds (75596 ms)
[ec2-user@ip-172-31-22-181 pig-0.15.0]$
```

<u>Ans: 1min 15sec 596ms was taken</u>

Part 4: Hadoop Streaming

Implement query **0.3** using Hadoop streaming with python (you may implement this part in Java if you prefer).
SELECT lo_quantity, SUM(lo_revenue)
FROM lineorder
WHERE lo_discount < 3
GROUP BY lo_quantity;

Ans:
Python Code (Mapper): col8: lo_quantity, col12: lo_revenue, col11: lo_discount

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

for line in sys.stdin:
    words=line.strip()
    words=words.split('|')
    print '%s\t%s\t%s' % (words[8], words[12], words[11])
```

Python Code (Reducer): col0: lo_quantity, col1: lo_revenue, col2: lo_discount

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

qua=None #Preset the quantity is None
rev=None #Preset the revenue is None

for line in sys.stdin:
    words=line.strip()
    words=words.split('\t')

    if float(words[2]) < 3 :
        if qua == None:
            qua = words[0]
            rev = float(words[1])
        elif words[0] != qua :
            print '%s\t%s' % (qua, rev)
           qua = words[0]
           rev = float(words[1])
       elif words[0] == qua:
           rev += float(words[1])

print '%s\t%s' % (qua, rev)
```

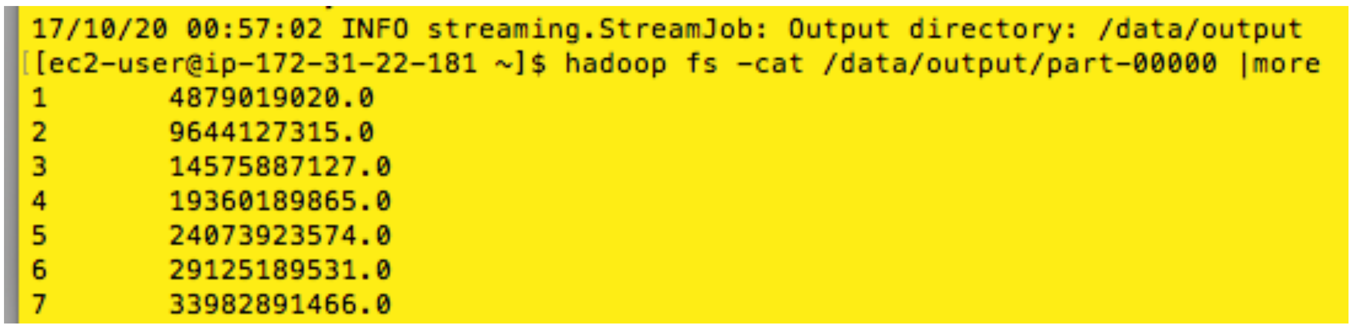
Testing in Terminal

```
[ec2-user@ip-172-31-22-181 ~]$ cat lineorder.tbl | python project_mapper1.py | sort -n | python
project_reducer1.py
```

Hadoop streaming Code

```
hadoop jar /home/ec2-user/hadoop-2.6.4/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -D
mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator -D
mapred.text.key.comparator.options=-n  -input /projectData -output /data/output  -mapper
project_mapper1.py  -reducer project_reducer1.py -file project_reducer1.py  -file
project_mapper1.py
```

Output:



```
17/10/20 00:57:02 INFO streaming.StreamJob: Output directory: /data/output
[ec2-user@ip-172-31-22-181 ~]$ hadoop fs -cat /data/output/part-00000 |more
1       4879019020.0
2       9644127315.0
3       14575887127.0
4       19360189865.0
5       24073923574.0
6       29125189531.0
7       33982891466.0
```

<u>Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Project Phase 1" at the top.</u>