**Ronaldlee Ejalu**

CSC 555 Project Phase 1

In this part of the project (which will serve as our take-home midterm), you will 1) Set up a 3-node cluster and 2) perform data warehousing and transformation queries using Hive, Pig and Hadoop streaming on that cluster. The modified Hive-style schema is:

<http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql>

(**you still have to add the delimiter to table definitions**)

It is based on SSBM benchmark (derived from industry standard TPCH benchmark). The data is at 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.

[http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/dwdate.tbl](%20http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/dwdate.tbl)

[http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/lineorder.tbl](%20http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/lineorder.tbl)

[http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/part.tbl](%20http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/part.tbl)

[http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/supplier.tbl](%20http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/supplier.tbl)

[http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/customer.tbl](%20http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/customer.tbl)

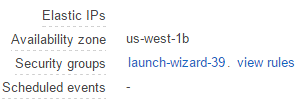
Please be sure to submit all code (pig, python and HiveQL).

# Part 1: Multi-node cluster

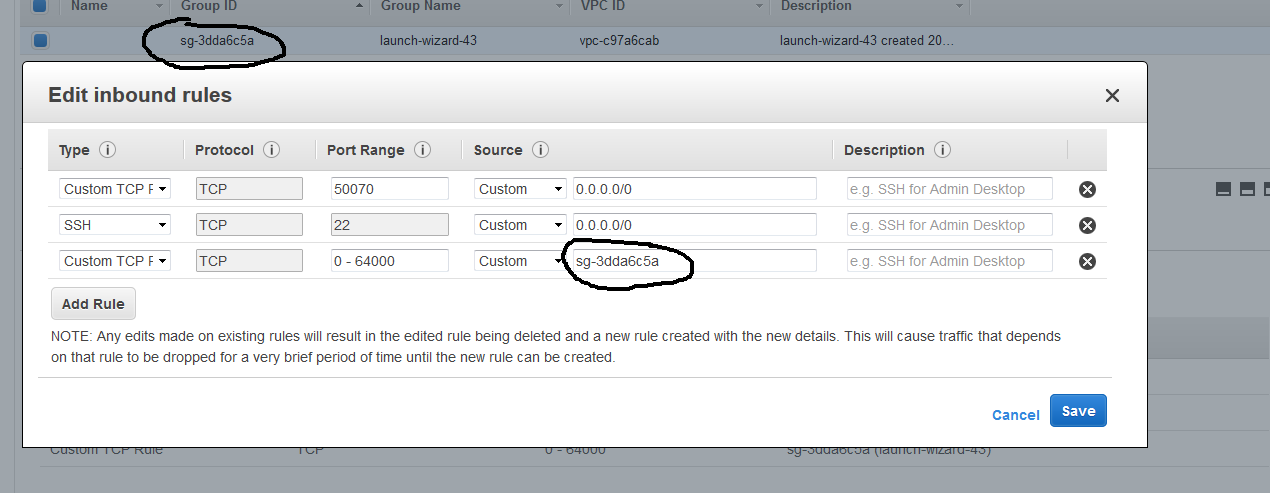
1. Your first step is to setup a multi-node cluster and re-run wordcount. For this part, you will create a 3-node cluster (with a total of 1 master + 2 worker nodes). Include your master node in the workers file, to make sure **all 3** nodes are working.

You need to perform the following steps:

1. Create a medium machine on AWS (which will serve as your master). It is possible, but I do not recommend trying to reconfigure your existing Hadoop setup into this new cluster (it is much easier to make 3 new nodes for a total of 4).
   1. When creating a node I recommend changing the default 8G hard drive to 20G.
   2. Change your security group setting to open firewall access. We need to open the ports in two different ways. We will open port 50070 for the web interface in order to be able to see the cluster status in a browser. We will also set 0-64000 range opening up all ports. However, we will ensure that the ports are open only **within** the cluster and not to the world.

In order to make changes, you need to do the following. Access the cluster security group (launch-wizard-xx). Right click on the security group and choose Edit inbound rules

Note that the first line below is opening port 50070. The second line below is the default (port 22 is required for regular SSH connections). The third line opens all ports but ONLY for the same security group (assuming that all of your nodes in the cluster share the same security group). Please note that we previously had some issues with machines being hacked without that last limitation, so please don’t skip this step



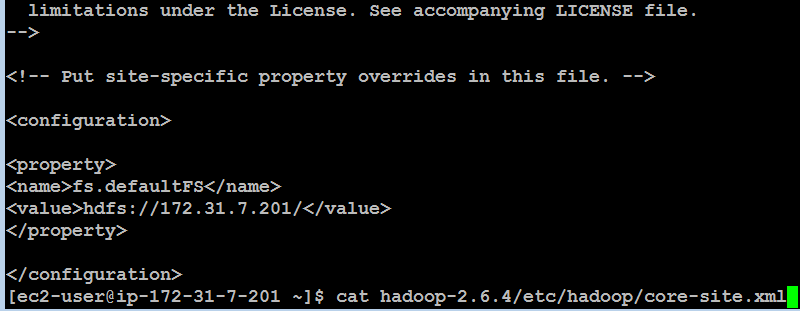
* 1. Create two new small machines and make sure they are using the same security group that you have configured on the master. You would need to change the security group settings so that both of the workers are sharing the same security group. For that, you can go to “Networking”, “Change Security Groups” and check the security group you want.

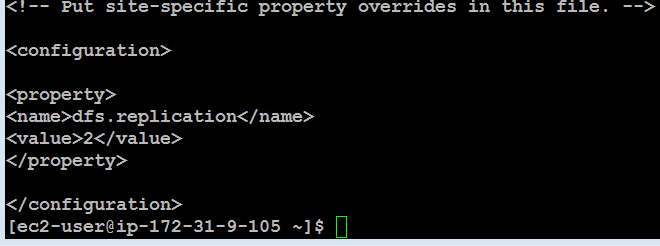
NOTE: Please make sure to label the machines so that they are easy to find, as it may get a little cluttered.

1. Connect to the master and set up Hadoop similarly to what you did previously. Use the following link:

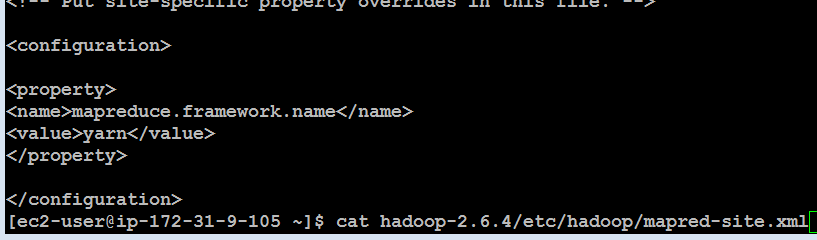
<http://dbgroup.cdm.depaul.edu/Courses/CSC555/hadoop-2.6.4.tar.gz>

Do not set up Hadoop on the workers – you will only need to configure up Hadoop once.

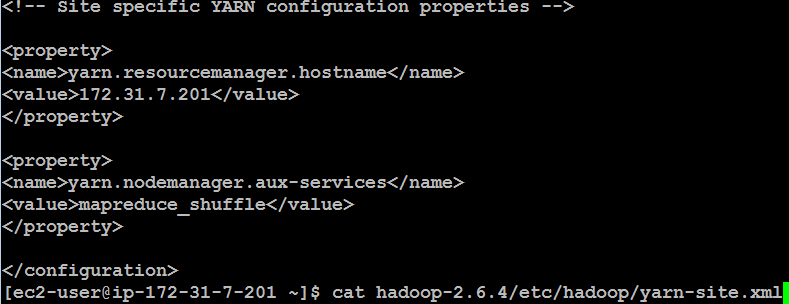
* 1. Configure core-site.xml, adding the **PrivateIP** (do not use public IP) of the master. 
  2. Configure hdfs-site and set replication factor to 2.



* 1. 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



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



Finally, edit the workers 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/workers

172.31.7.201

172.31.5.246

…

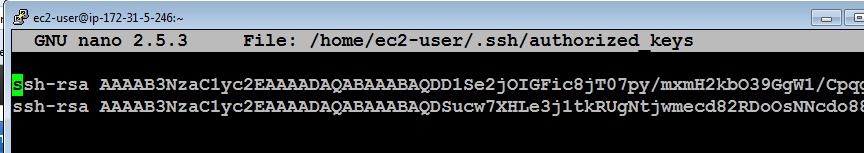
Make sure that you use private IP (private DNS is also ok) for your configuration files (such as conf/masters and conf/workers or the other 3 config files). The advantage of the Private IP is that it does not change after your instance is stopped (if you use the Public IP, the cluster would need to be 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 (like discussed in class) would prevent the key from matching it’s private key pair.

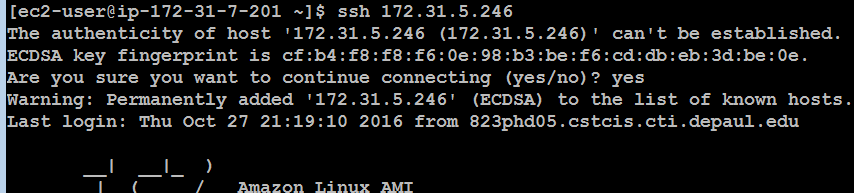
Note that the example below is NOT the master, but one of the workers (ip-172-31-5-246). The first public key is the .pem Amazon half and the 2nd public key is the master’s public key copied in as one line.



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.



Once you 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 the contents of the hadoop directory there. Make sure your Hadoop installation is configured correctly (because from now on, you will have 4 copies of the Hadoop directory and all changes need to be applied in 4 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)

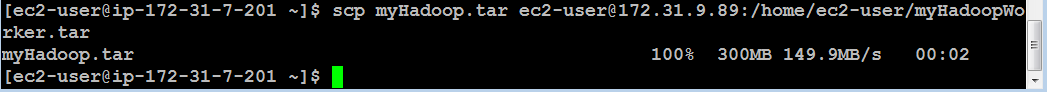
tar cvf myHadoop.tar hadoop-2.6.4

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



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

You also need to install Java using **sudo yum install ant**. Without Java on the worker nodes, Hadoop will not start.

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 yet successfully copied myHadoopWorker.tar file to the worker.

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

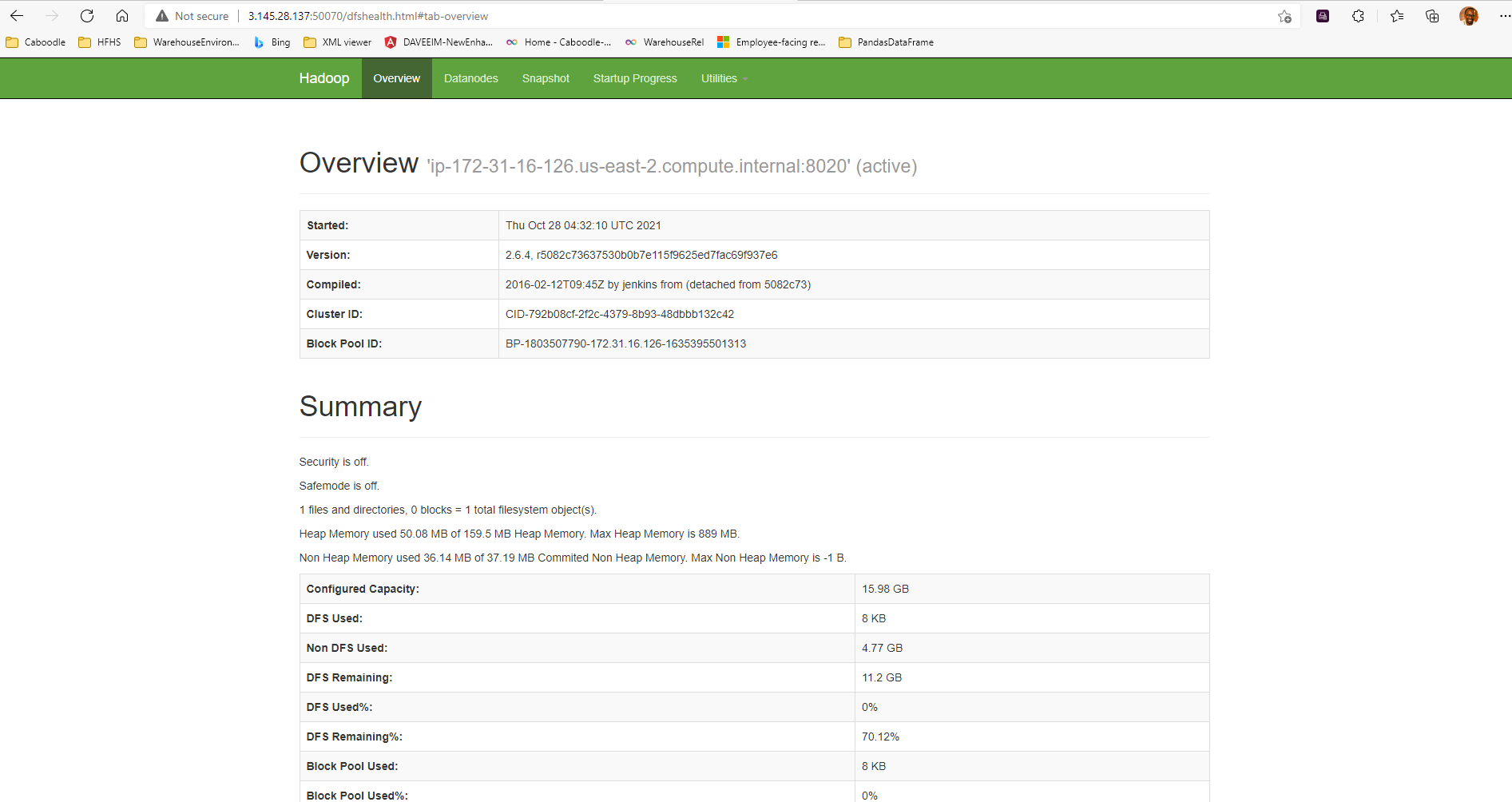
Once you have successfully completed the previous steps, you should can start and use your new cluster by going to the master node and running the start-dfs.sh and start-yarn.sh scripts (you do not need to explicitly start anything on worker nodes – the master will do that for you).

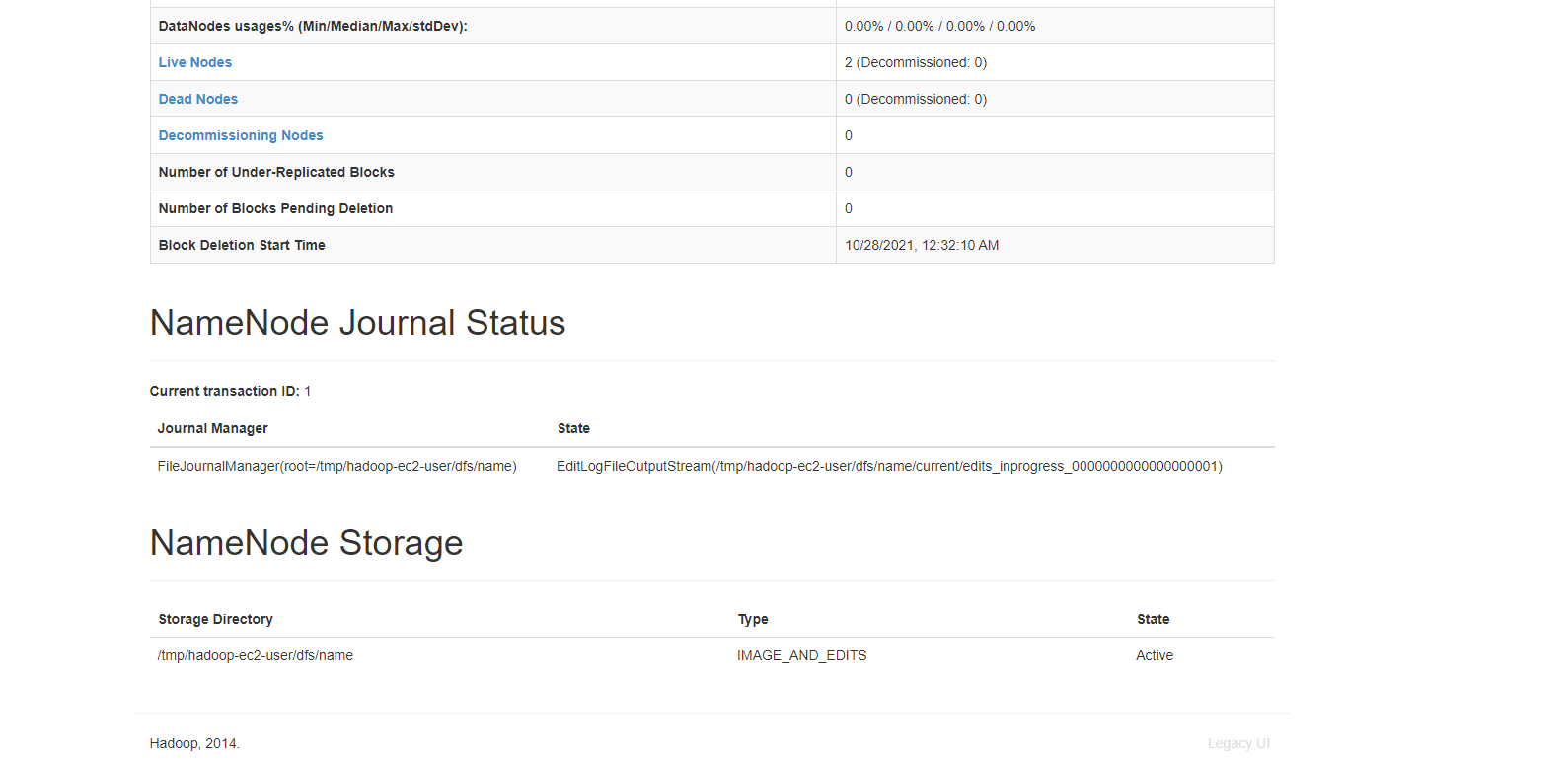
You should verify that the cluster is running by pointing your browser to the link below.

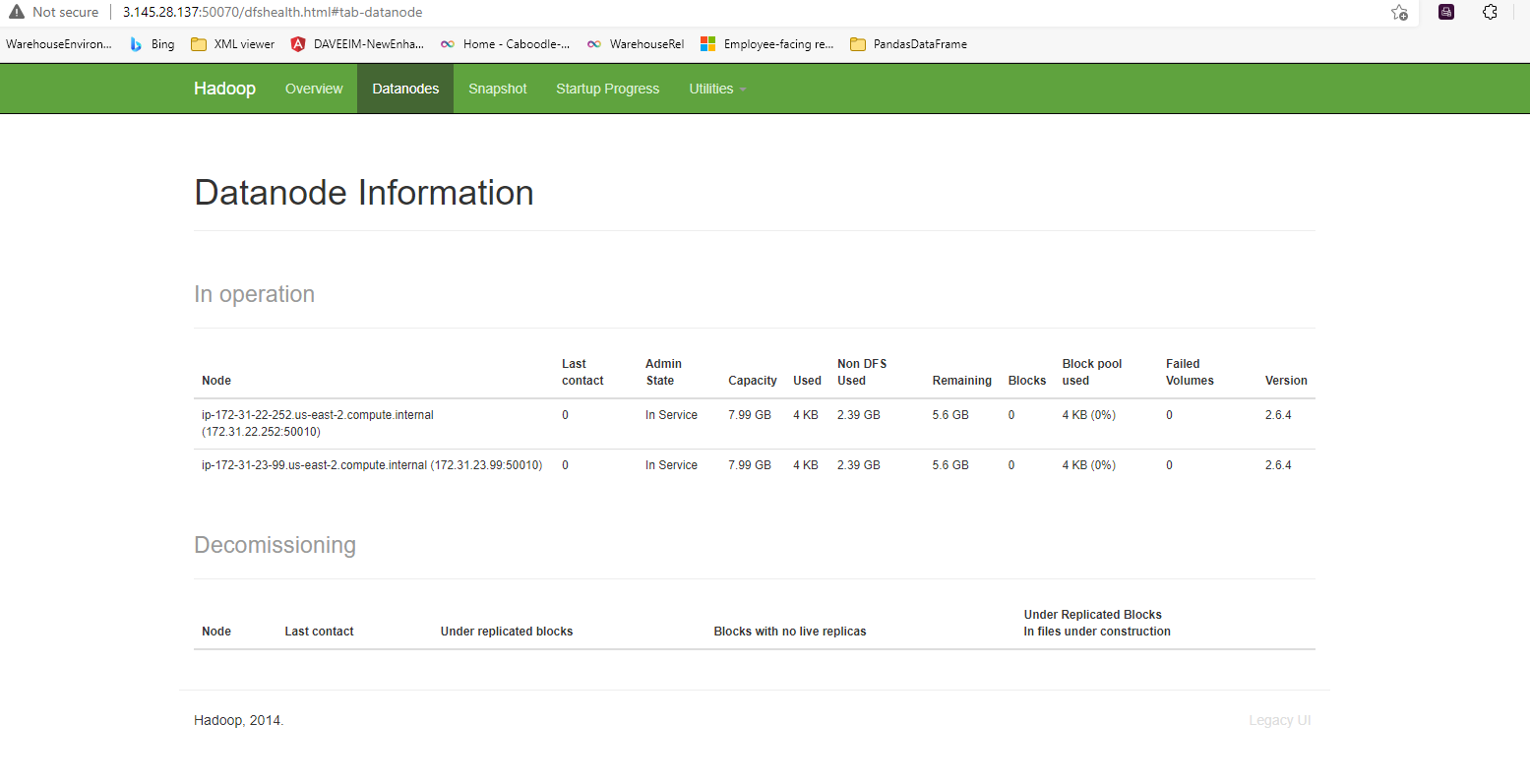
http://[insert-the-public-ip-of-master]:50070/

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

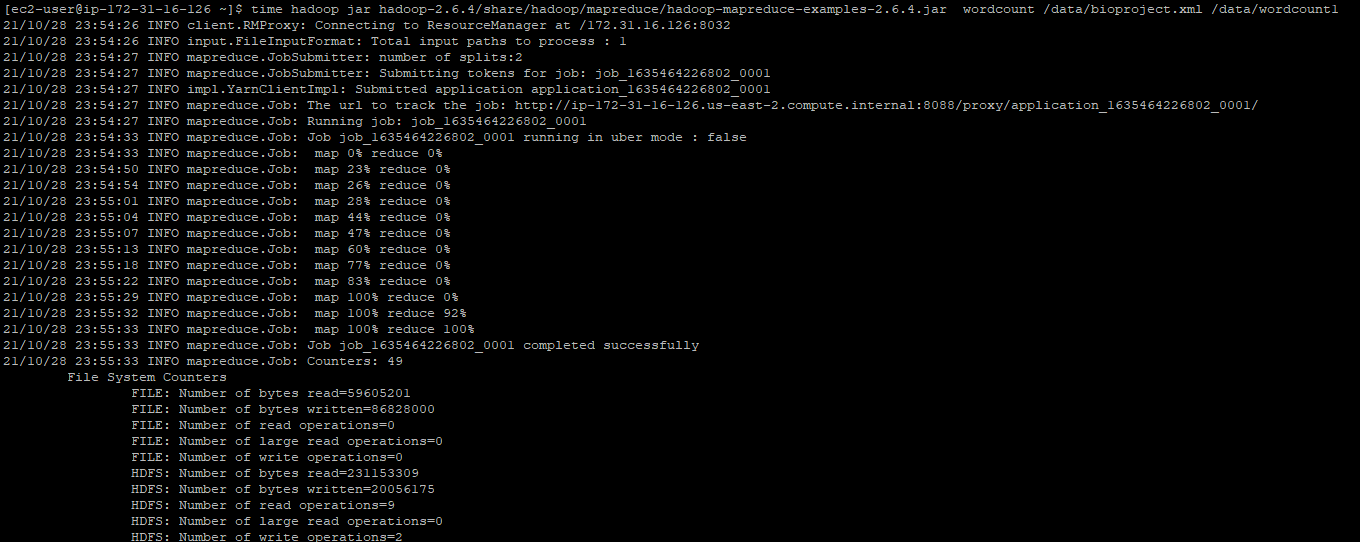
Submit a screenshot of your cluster status view.

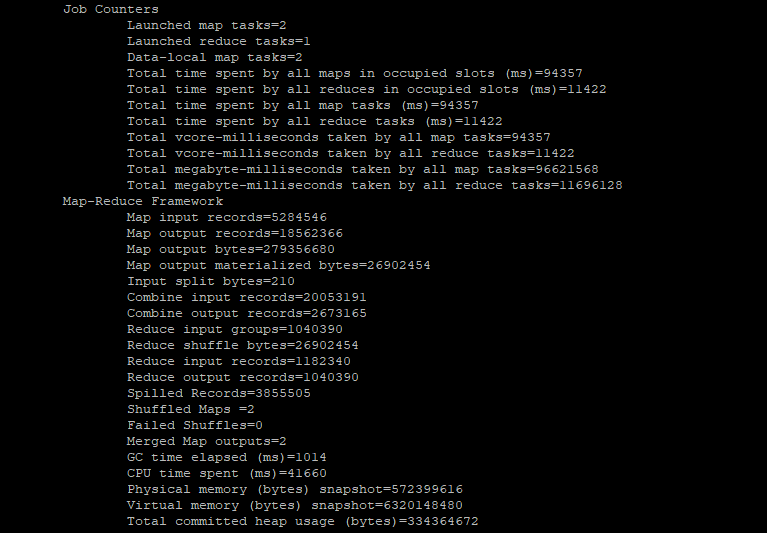


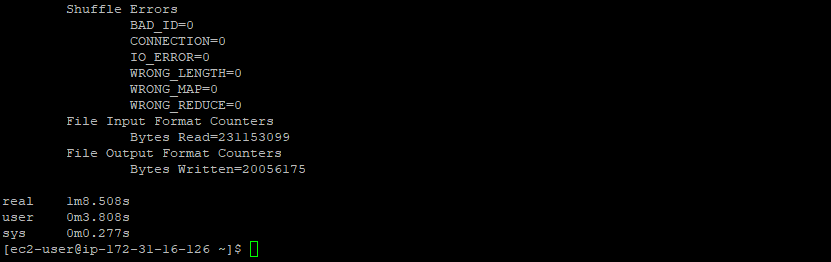




Repeat the steps for wordcount using bioproject.xml from Assignment 2 and submit screenshots of running it.







Submit a short paragraph with a discussion about how the results compare (faster? slower? How much faster/slower?)

The word count on the cluster runs faster than the one in assignment 2, which we ran on a single cluster. There is a difference of 4 mins in the run time.

# Part 2: Hive

1. Run the following query in Hive and report the time it takes to execute:

select lo\_orderdate, sum(lo\_extendedprice) as revenue

from lineorder, dwdate

where lo\_orderdate = d\_datekey

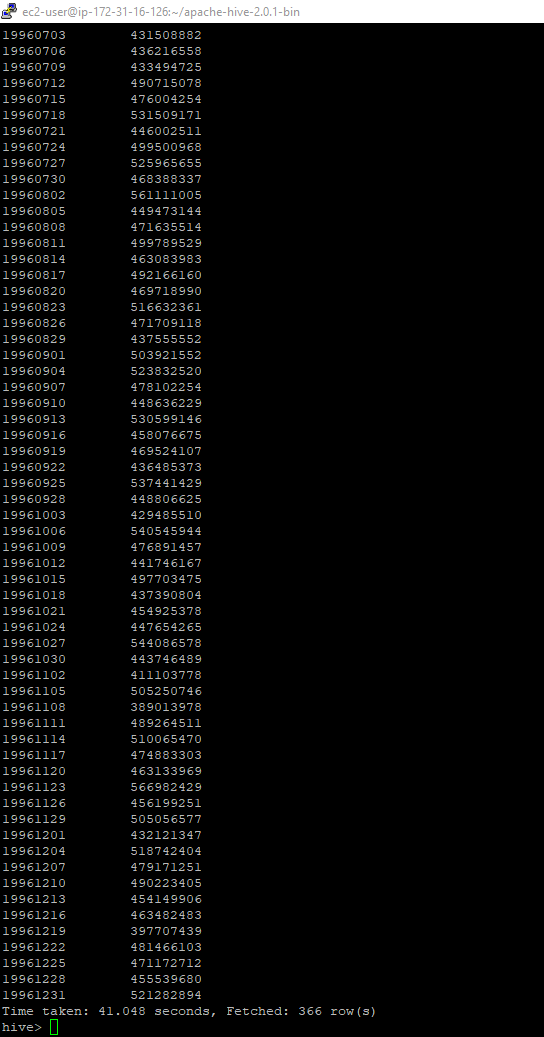
and d\_year = 1996

and lo\_discount between 4 and 6

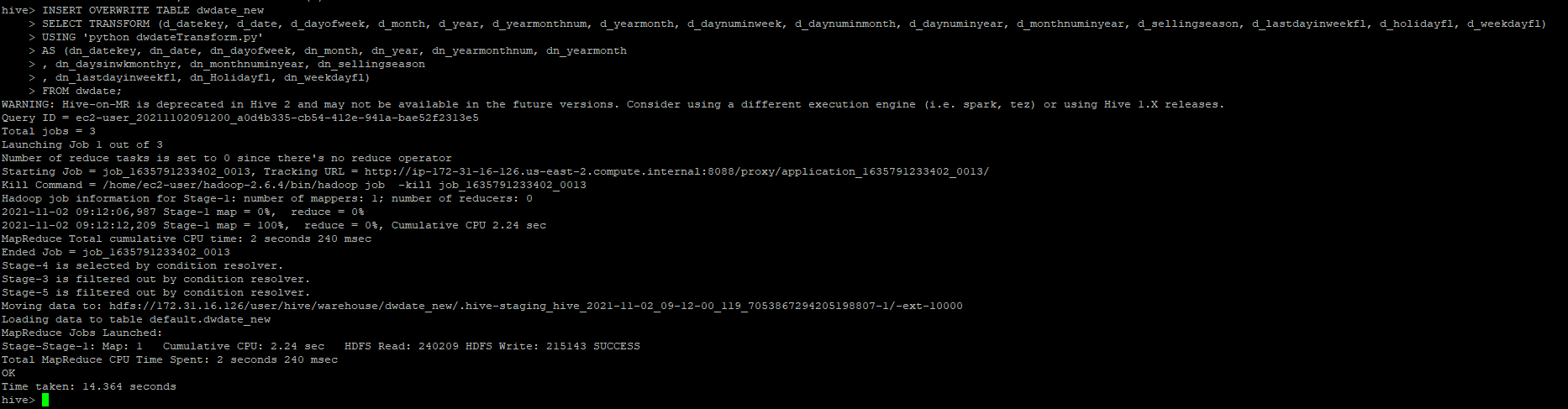
and lo\_quantity < 22

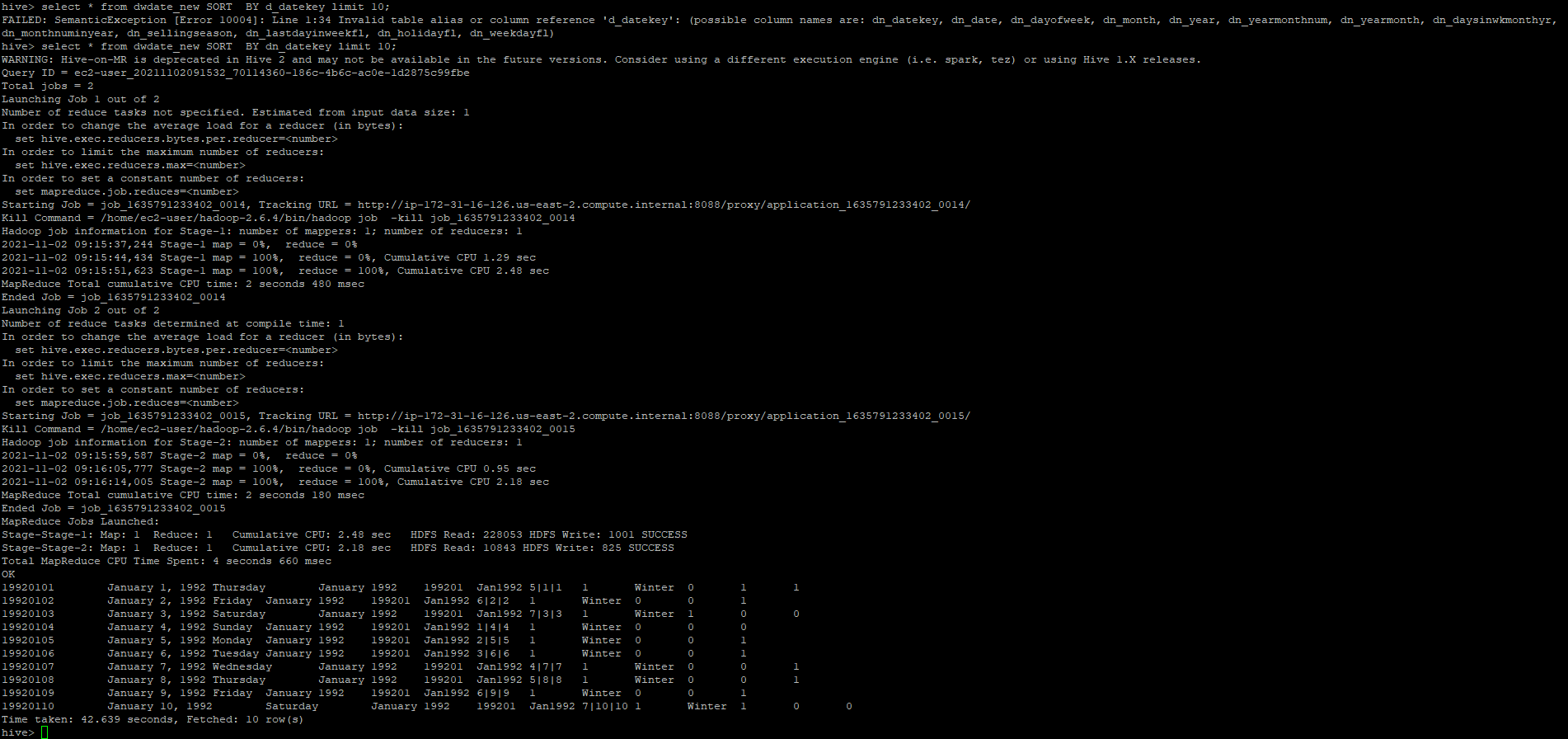
GROUP BY lo\_orderdate;

It takes 41.048 seconds to execute as indicated below:



1. Perform the following transform operation using SELECT TRANSFORM on the dwdate table by creating a new table. The new dwdate table will combine d\_daynuminweek, d\_daynuminmonth, and d\_daynuminyear into a single column in the new table using a delimiter of your choice. You should also eliminate the following 2 columns: d\_lastdayinmonthfl and d\_weeknuminyear. The final table will have fewer columns than the original table because you merge 3 columns into 1 and remove 2 columns.





Hive scripts:



dwdateTransform.py



# Part 3: Pig

Convert and load the data into Pig, implementing and timing the following queries:

SELECT lo\_discount, AVG(lo\_extendedprice)

FROM lineorder

GROUP BY lo\_discount;

SELECT lo\_quantity, SUM(lo\_revenue)

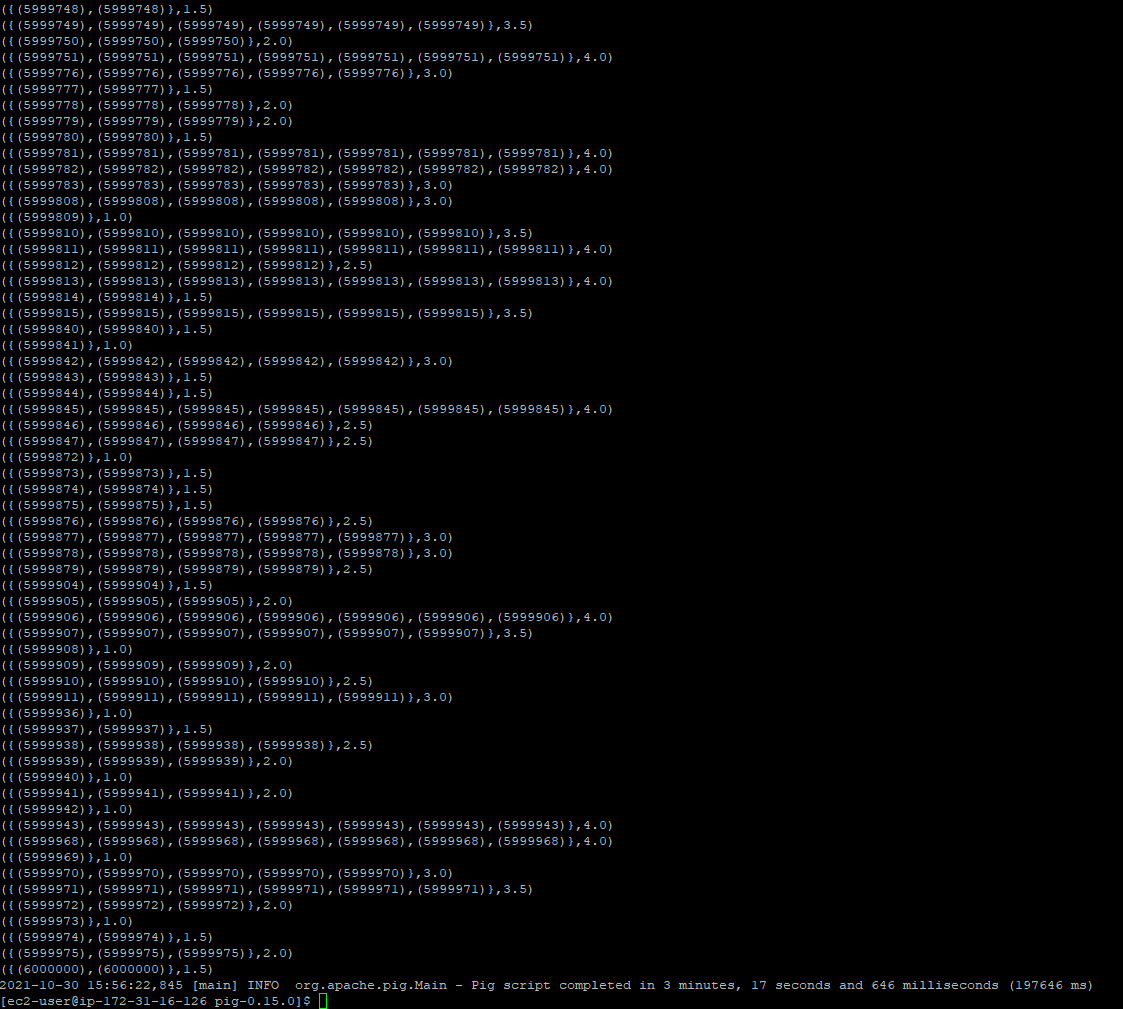
FROM lineorder

WHERE lo\_discount > 8 AND lo\_quantity > 33

GROUP BY lo\_quantity;

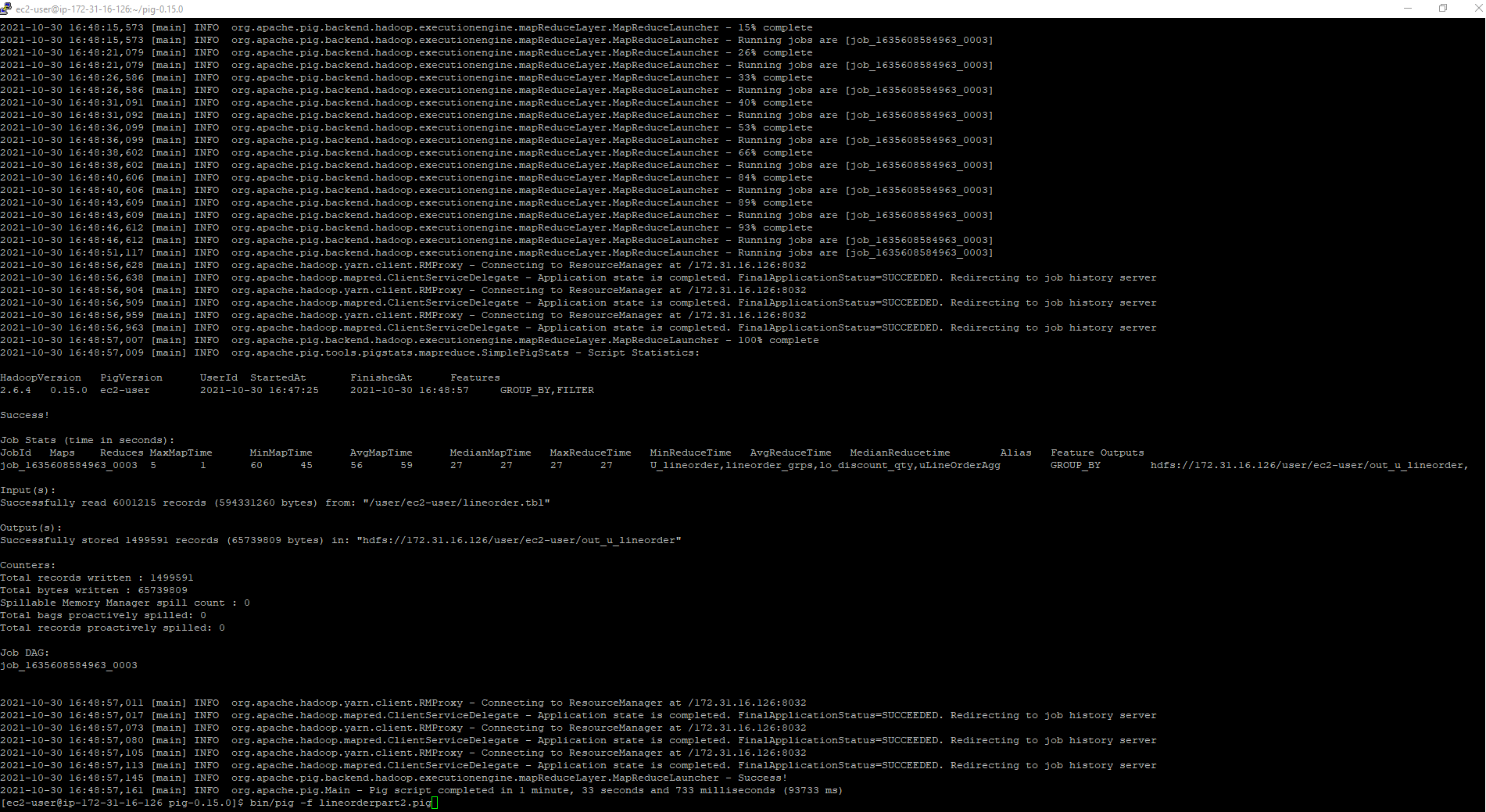
One easy way to time Pig is as follows: put your sequence of pig commands, including LOAD, into a text file and then run, from command line in pig directory (e.g., [ec2-user@ip-172-31-6-39 pig-0.15.0]$), bin/pig -f pig\_script.pig (which will report how long the pig script took to run).

My first script completed in 3 minutes, 17 seconds and 646 milliseconds as indicated below in the screen shot.

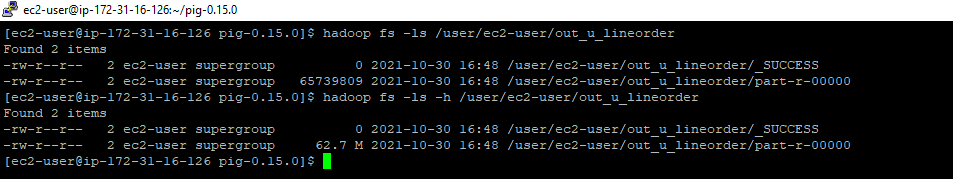


Store the results of the 2nd Pig query into HDFS and report the size of the output.

The second Pig script (lineorderpart2.pig), completed in 1 minute, 33 seconds and 733 milliseconds as indicated below in the screen shot:



The size of the output file is: 62.7 M as shown below in the screen shot:



PigScripts:





# Part 4: Hadoop Streaming

Implement, run and time the following query using Hadoop streaming with python.

SELECT lo\_quantity, MAX(lo\_revenue)

FROM (SELECT lo\_revenue, MAX(lo\_quantity) as lo\_quantity,

MAX(lo\_discount) as lo\_discount

FROM lineorder

WHERE lo\_orderpriority LIKE '%URGENT'

GROUP BY lo\_revenue)

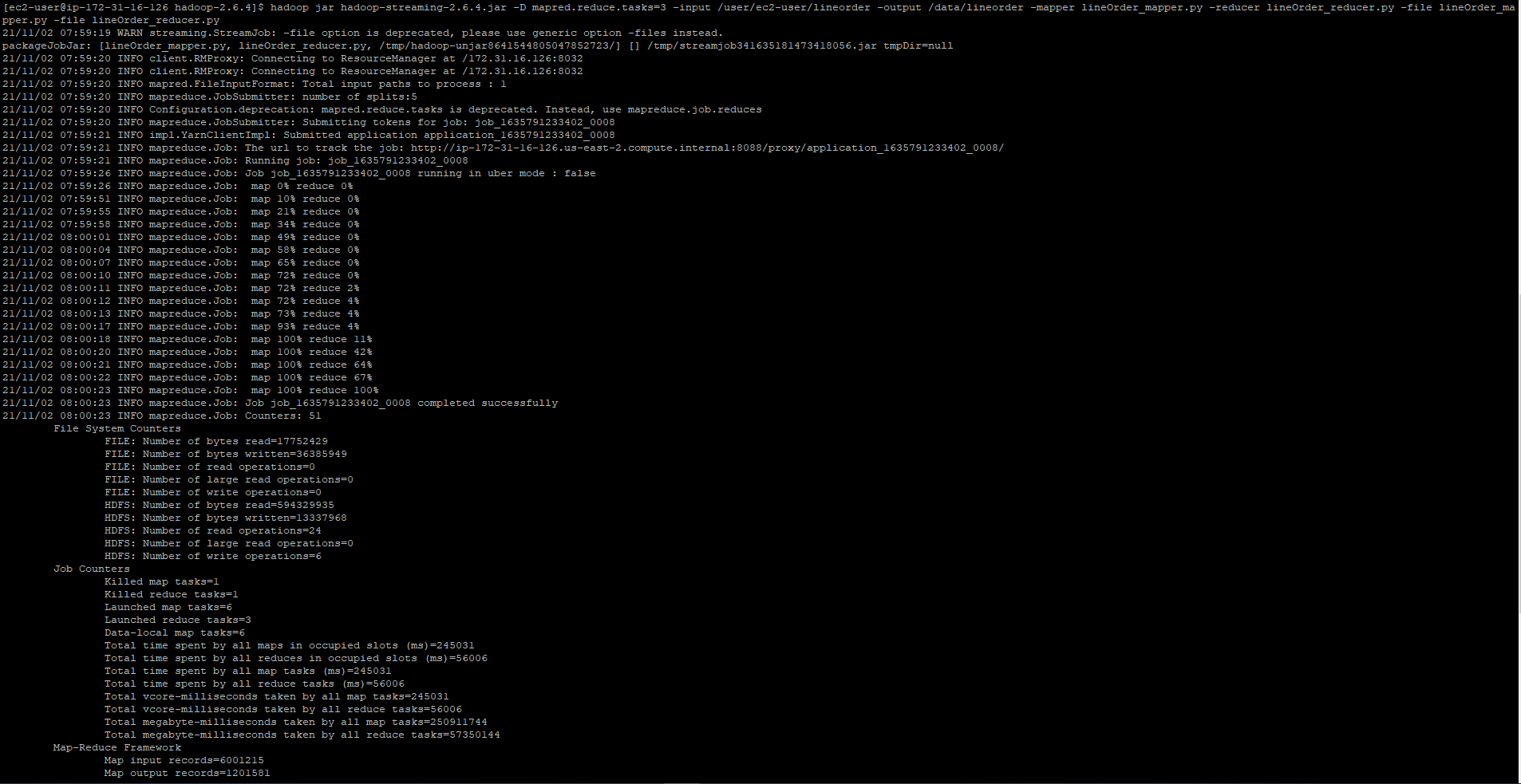
WHERE lo\_discount BETWEEN 4 AND 8

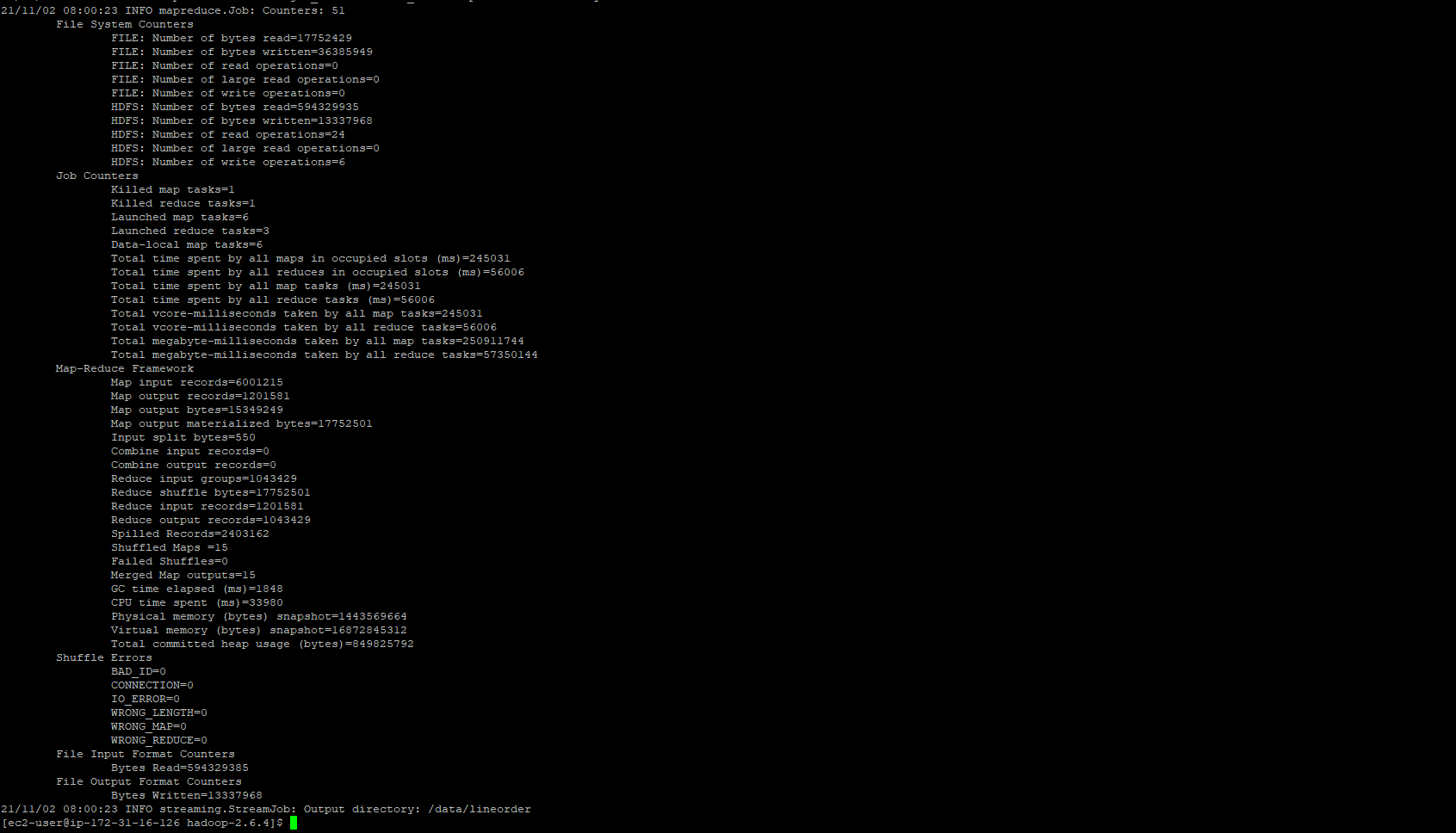
GROUP BY lo\_quantity;

This requires running two different map reduce jobs. First, you would write a job that executes the subquery and produces an output in HDFS. Then you would write a second job that uses output of the first job as the input.

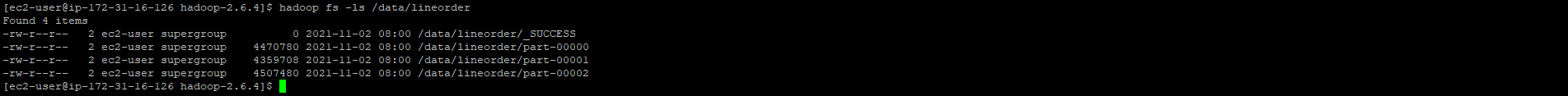
Don’t forget to submit your python code, and the command line you used to run Hadoop streaming jobs.

The first job command lines:

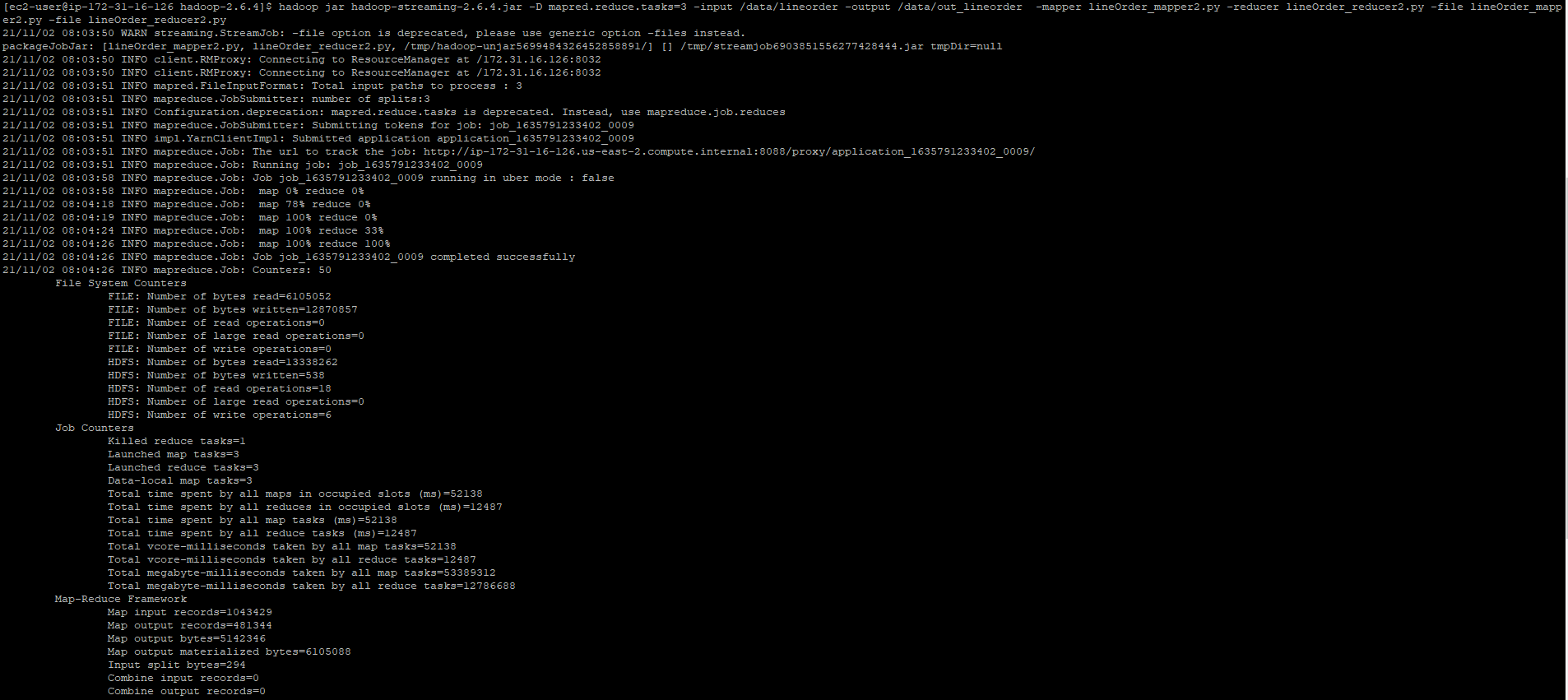


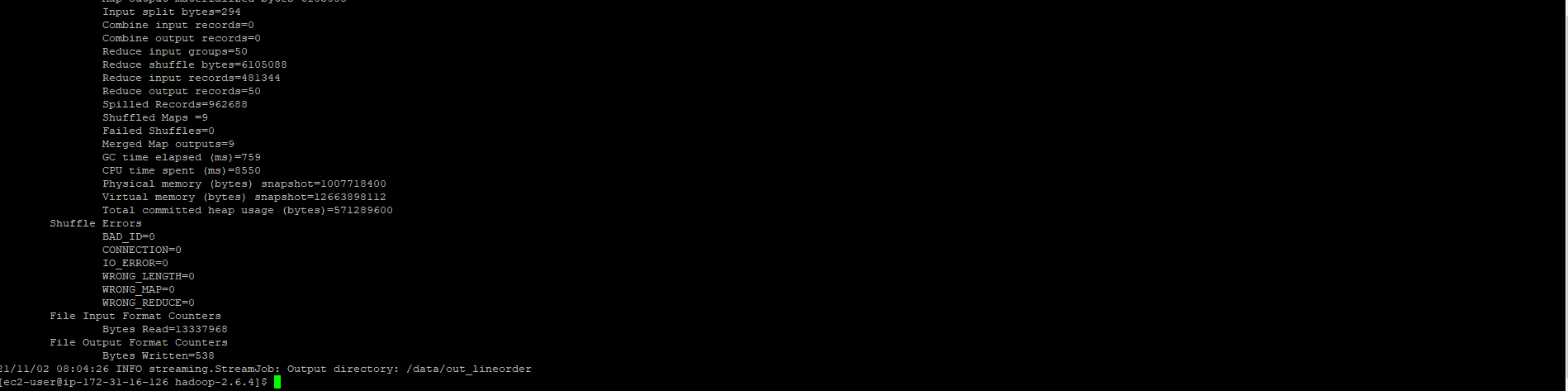


The output files from the first job:

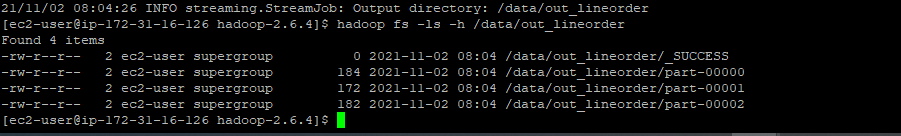


The commands from the second job are as follows:





The output files from the second job:



**Code for the first job:**

lineOrder\_mapper.py



lineOrder\_reducer.py



Python Code for the second job:

lineOrder\_mapper2.py



lineOrder\_reducer2.py



NOTE: You may implement this part in Java if you prefer.

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.