CSC 555 Mining Big Data

Project, Phase 1

Lavinia Wang #1473704

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

Part 1: Multi-node cluster

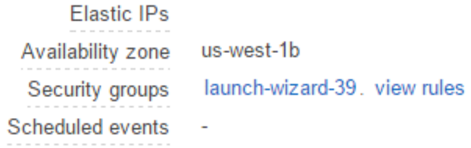
1) Your first step is to setup a multi-node cluster and re-run a simple wordcount. For this part, you will create a 3-node cluster (with a total of 1 master + 2 worker nodes). Include your master node in the "slaves" file, to make sure **all 3** nodes are working.
You need to perform the following steps:

   1. Create a new node of a medium size (you can always switch the size of the node). 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 on all nodes.**

      b. 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.

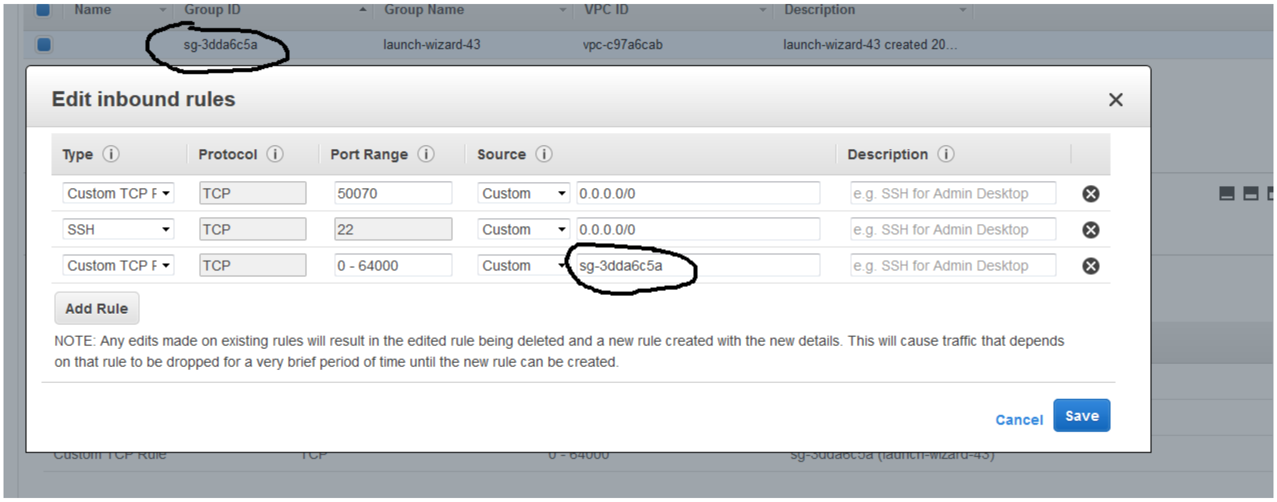
In order to make changes, you need to do the following. Access the cluster security group (launch-wizard-xx).



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

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 – that will happen automatically if you use the "create more like this" option when creating instances as specified in part 1-c below). We previously had some issues with machines being hacked without that last limitation, so make sure you include it.



Name	Group ID	Group Name	VPC ID	Description
	sg-3dda6c5a	launch-wizard-43	vpc-c97a6cab	launch-wizard-43 created 20…

Edit inbound rules

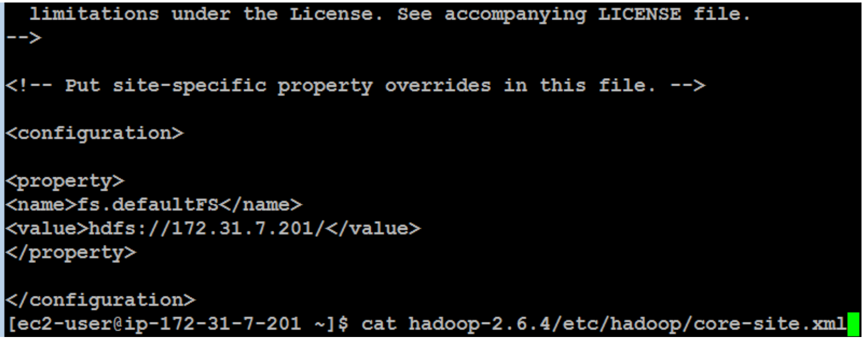
Type	Protocol	Port Range	Source		Description
Custom TCP F	TCP	50070	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	0 - 64000	Custom	sg-3dda6c5a	e.g. SSH for Admin Desktop

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

c. 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.

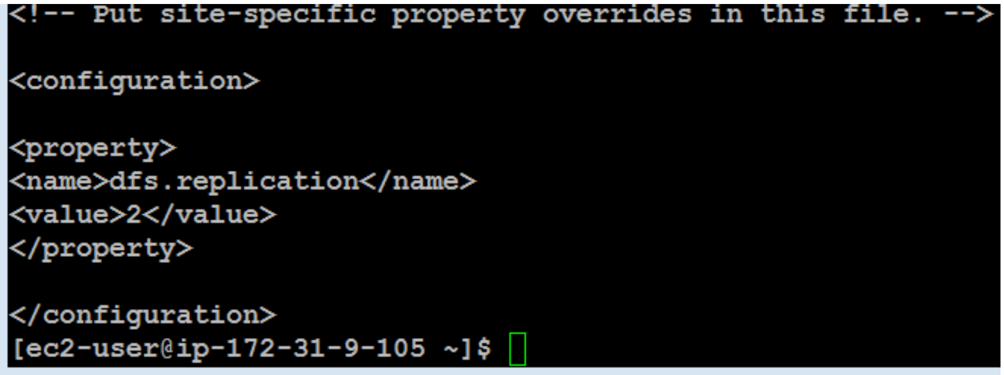
2. Connect to the master and set up Hadoop similarly to what you did previously. Do not attempt to repeat these steps on workers yet – you will only need to set up Hadoop once.

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



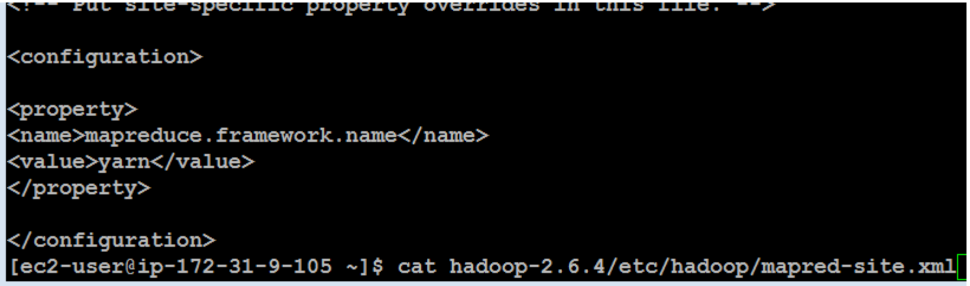
```
  limitations under the License. See accompanying LICENSE file.
-->

<!-- 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.



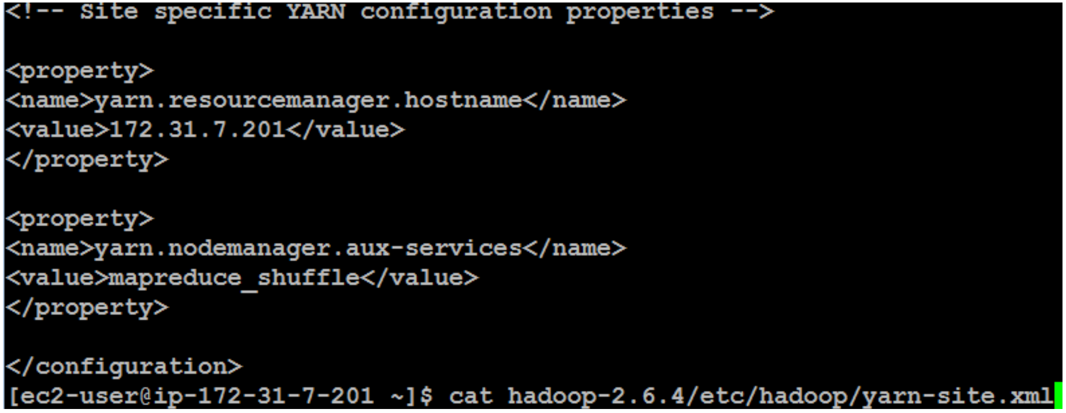
```
<!-- Put site-specific property overrides in this file. -->

<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 4 nodes (master and 3 workers) using Private IPs

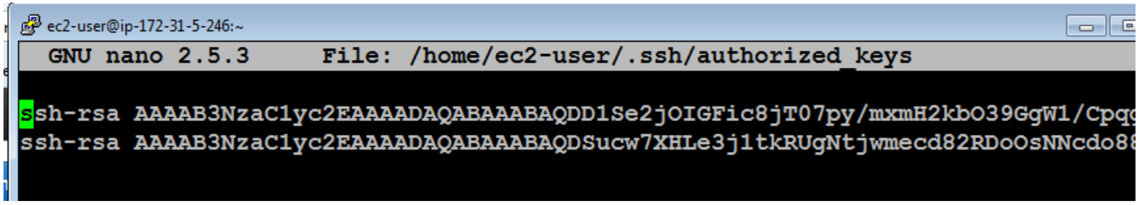
```
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/slaves
172.31.7.201
172.31.5.246
…
```

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.

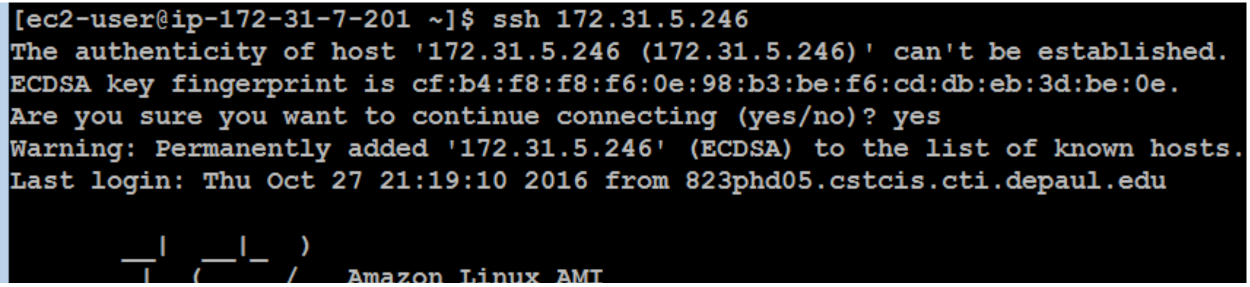
1. Run ssh-keygen -t rsa (and enter empty values for the passphrase) on the master node. That will generate .ssh/id_rsa and .ssh/id_rsa.pub (private and public key). You now need to manually copy the .ssh/id_rsa.pub and append it to ~/.ssh/authorized_keys **on each worker.**
   Keep in mind that this is a single-line public key and accidentally introducing a line break would cause a mismatch.
   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.

   ```
     GNU nano 2.5.3      File: /home/ec2-user/.ssh/authorized_keys

   ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDD1Se2jOIGFic8jT07py/mxmH2kbO39GgW1/Cpq
   ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSucw7XHLe3j1tkRUgNtjwmecd82RDoOsNNcdo8
   ```

   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
```

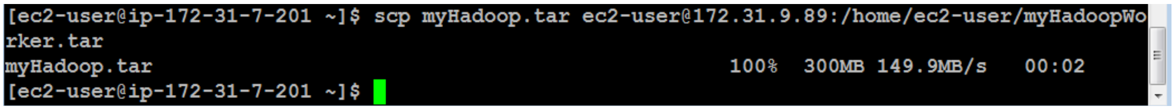
Once you have verified that you can ssh from the master node to every cluster member including the master itself (ssh localhost), you are going to return to the master node (**exit** until your prompt shows the IP address of the master node) and pack the contents of the hadoop directory there. Make sure 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



```
[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 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).

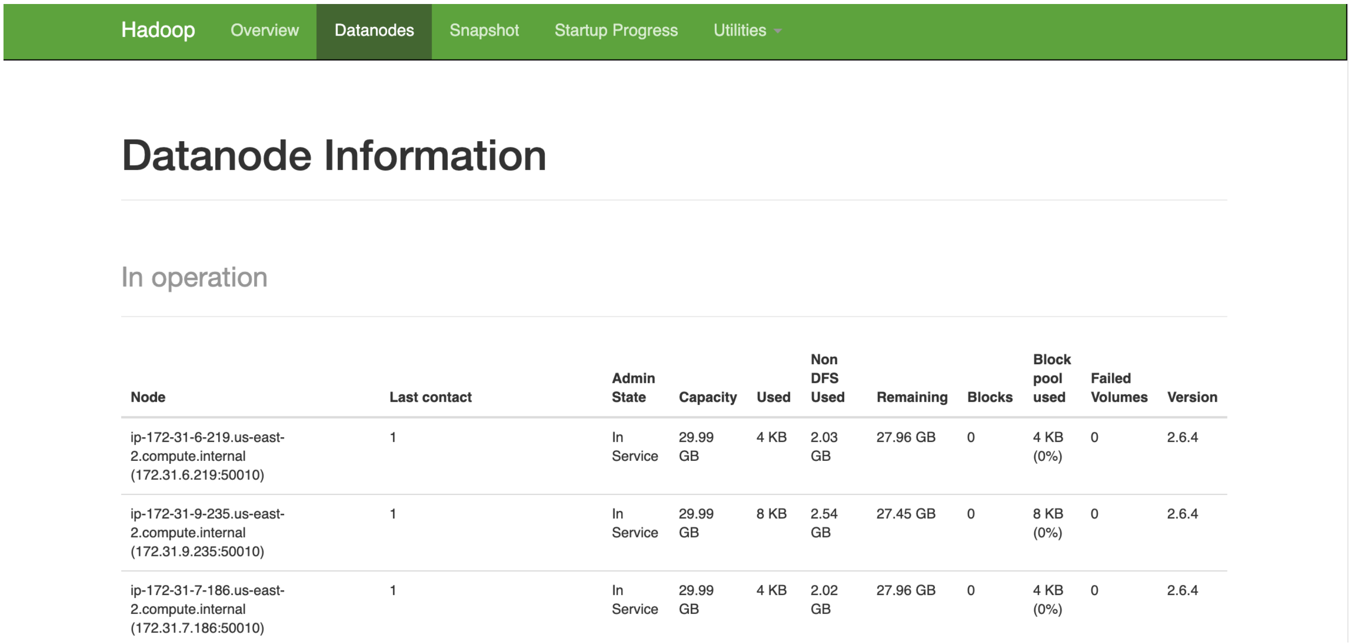
You should verify that the cluster is running by pointing your browser to the link below.

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

Make sure that the cluster is operational (you can see the 4 nodes under Datanodes tab).

Submit a screenshot of your cluster status view.

Lavinia Wang #1473704

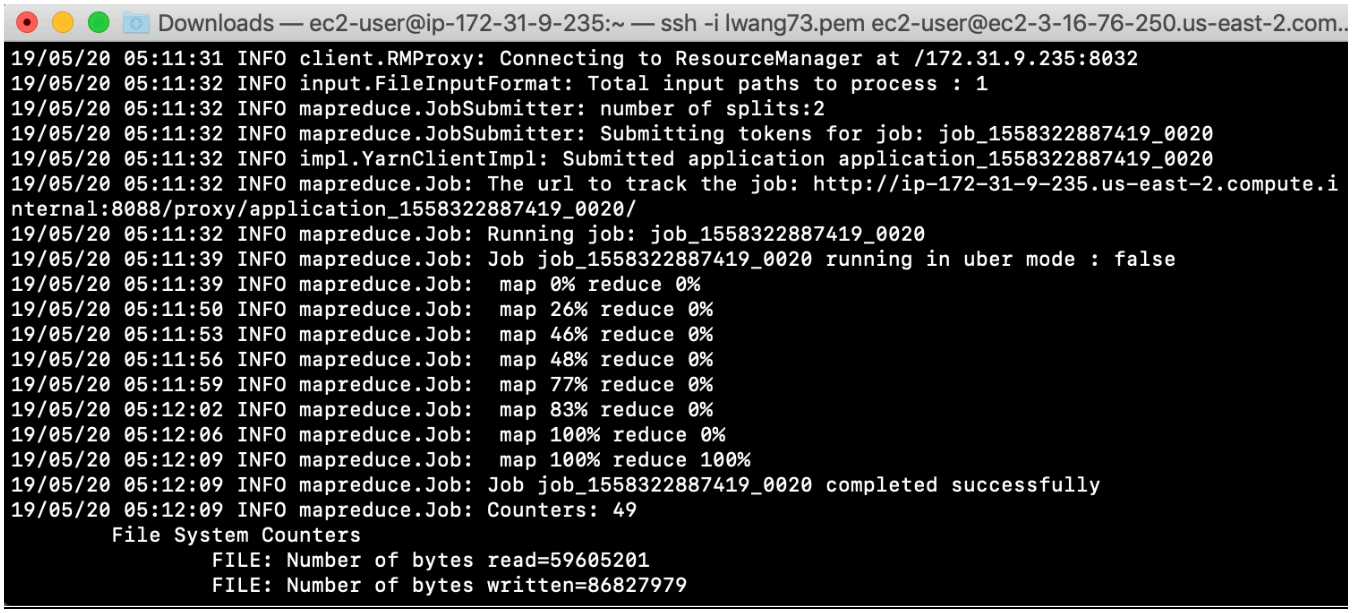


Datanode Information

In operation

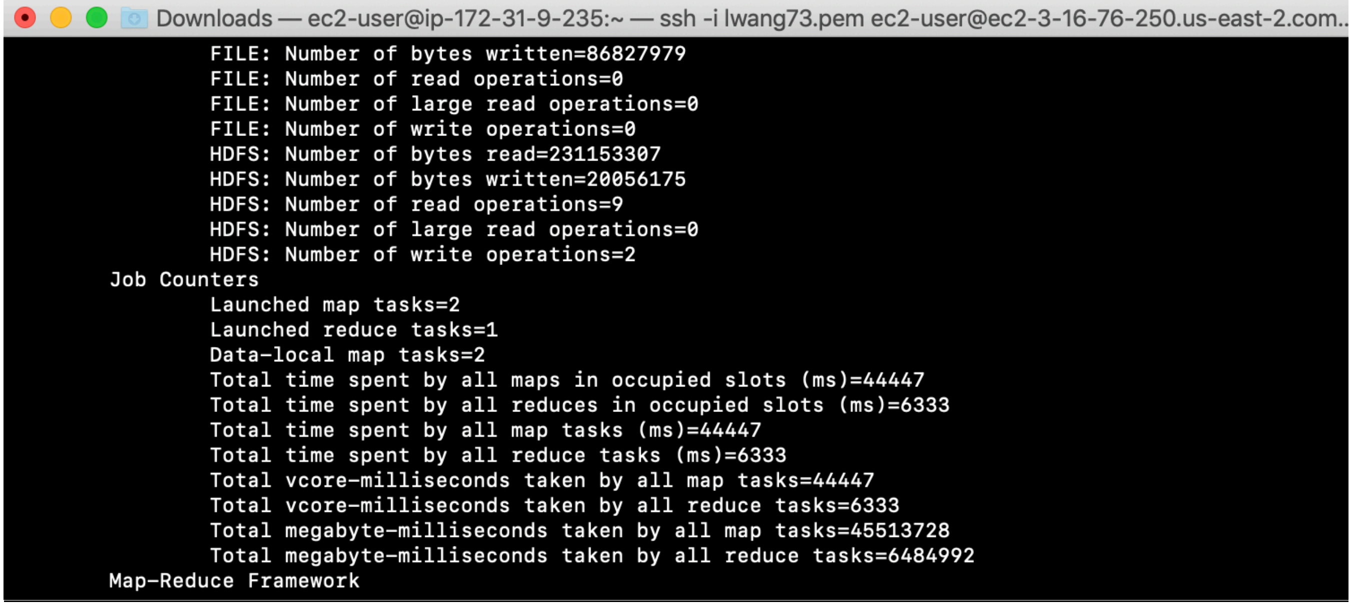
Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-6-219.us-east-2.compute.internal (172.31.6.219:50010)	1	In Service	29.99 GB	4 KB	2.03 GB	27.96 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-9-235.us-east-2.compute.internal (172.31.9.235:50010)	1	In Service	29.99 GB	8 KB	2.54 GB	27.45 GB	0	8 KB (0%)	0	2.6.4
ip-172-31-7-186.us-east-2.compute.internal (172.31.7.186:50010)	1	In Service	29.99 GB	4 KB	2.02 GB	27.96 GB	0	4 KB (0%)	0	2.6.4

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

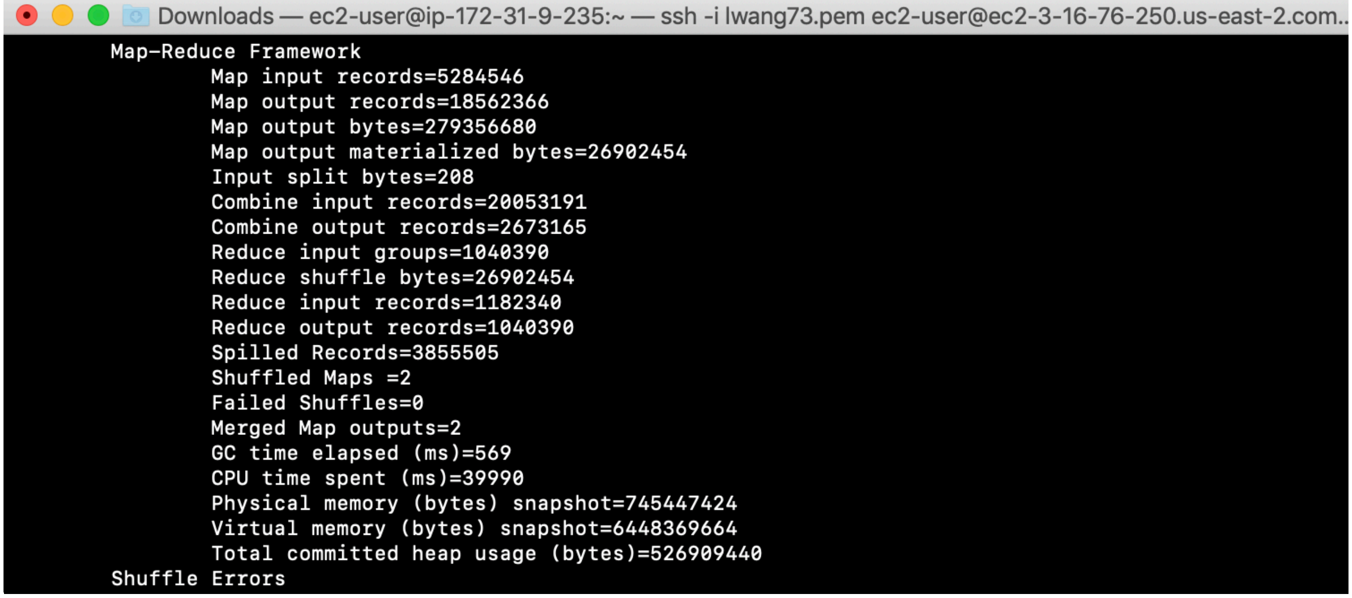

Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

```
19/05/20 05:11:31 INFO client.RMProxy: Connecting to ResourceManager at /172.31.9.235:8032
19/05/20 05:11:32 INFO input.FileInputFormat: Total input paths to process : 1
19/05/20 05:11:32 INFO mapreduce.JobSubmitter: number of splits:2
19/05/20 05:11:32 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1558322887419_0020
19/05/20 05:11:32 INFO impl.YarnClientImpl: Submitted application application_1558322887419_0020
19/05/20 05:11:32 INFO mapreduce.Job: The url to track the job: http://ip-172-31-9-235.us-east-2.compute.i
nternal:8088/proxy/application_1558322887419_0020/
19/05/20 05:11:32 INFO mapreduce.Job: Running job: job_1558322887419_0020
19/05/20 05:11:39 INFO mapreduce.Job: Job job_1558322887419_0020 running in uber mode : false
19/05/20 05:11:39 INFO mapreduce.Job:  map 0% reduce 0%
19/05/20 05:11:50 INFO mapreduce.Job:  map 26% reduce 0%
19/05/20 05:11:53 INFO mapreduce.Job:  map 46% reduce 0%
19/05/20 05:11:56 INFO mapreduce.Job:  map 48% reduce 0%
19/05/20 05:11:59 INFO mapreduce.Job:  map 77% reduce 0%
19/05/20 05:12:02 INFO mapreduce.Job:  map 83% reduce 0%
19/05/20 05:12:06 INFO mapreduce.Job:  map 100% reduce 0%
19/05/20 05:12:09 INFO mapreduce.Job:  map 100% reduce 100%
19/05/20 05:12:09 INFO mapreduce.Job: Job job_1558322887419_0020 completed successfully
19/05/20 05:12:09 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=59605201
                FILE: Number of bytes written=86827979
```

Lavinia Wang #1473704

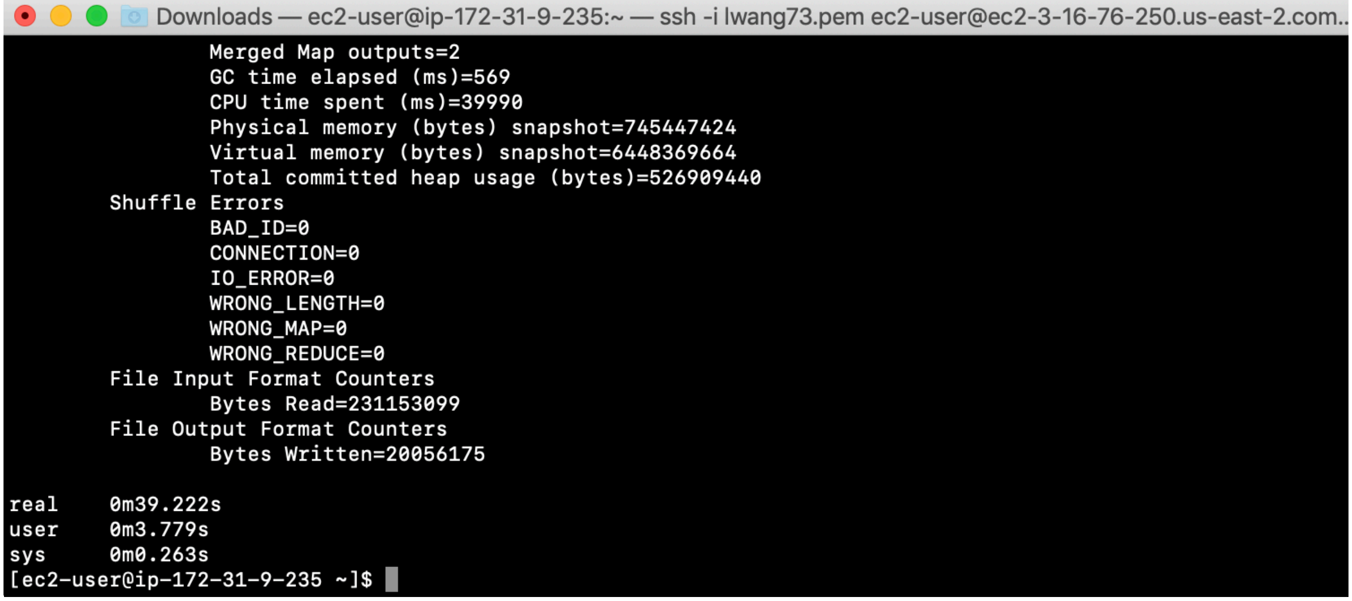

Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

```
                FILE: Number of bytes written=86827979
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=231153307
                HDFS: Number of bytes written=20056175
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=44447
                Total time spent by all reduces in occupied slots (ms)=6333
                Total time spent by all map tasks (ms)=44447
                Total time spent by all reduce tasks (ms)=6333
                Total vcore-milliseconds taken by all map tasks=44447
                Total vcore-milliseconds taken by all reduce tasks=6333
                Total megabyte-milliseconds taken by all map tasks=45513728
                Total megabyte-milliseconds taken by all reduce tasks=6484992
        Map-Reduce Framework
```


Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

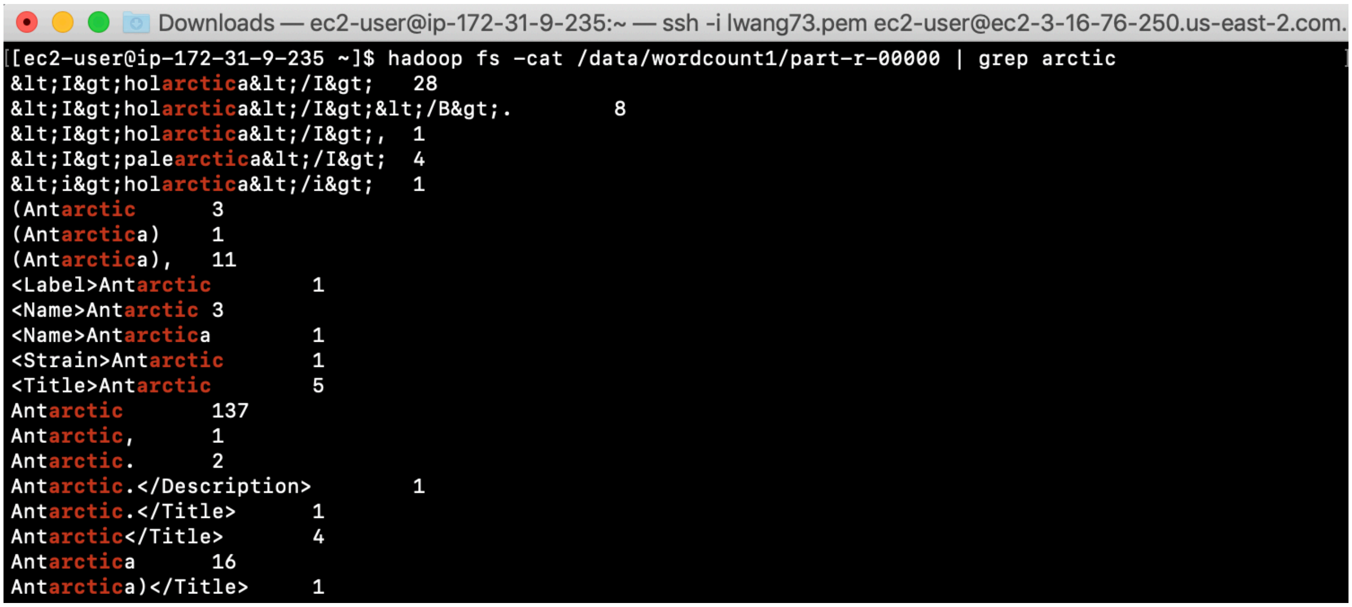
```
        Map-Reduce Framework
                Map input records=5284546
                Map output records=18562366
                Map output bytes=279356680
                Map output materialized bytes=26902454
                Input split bytes=208
                Combine input records=20053191
                Combine output records=2673165
                Reduce input groups=1040390
                Reduce shuffle bytes=26902454
                Reduce input records=1182340
                Reduce output records=1040390
                Spilled Records=3855505
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=569
                CPU time spent (ms)=39990
                Physical memory (bytes) snapshot=745447424
                Virtual memory (bytes) snapshot=6448369664
                Total committed heap usage (bytes)=526909440
        Shuffle Errors
```

Lavinia Wang #1473704


Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

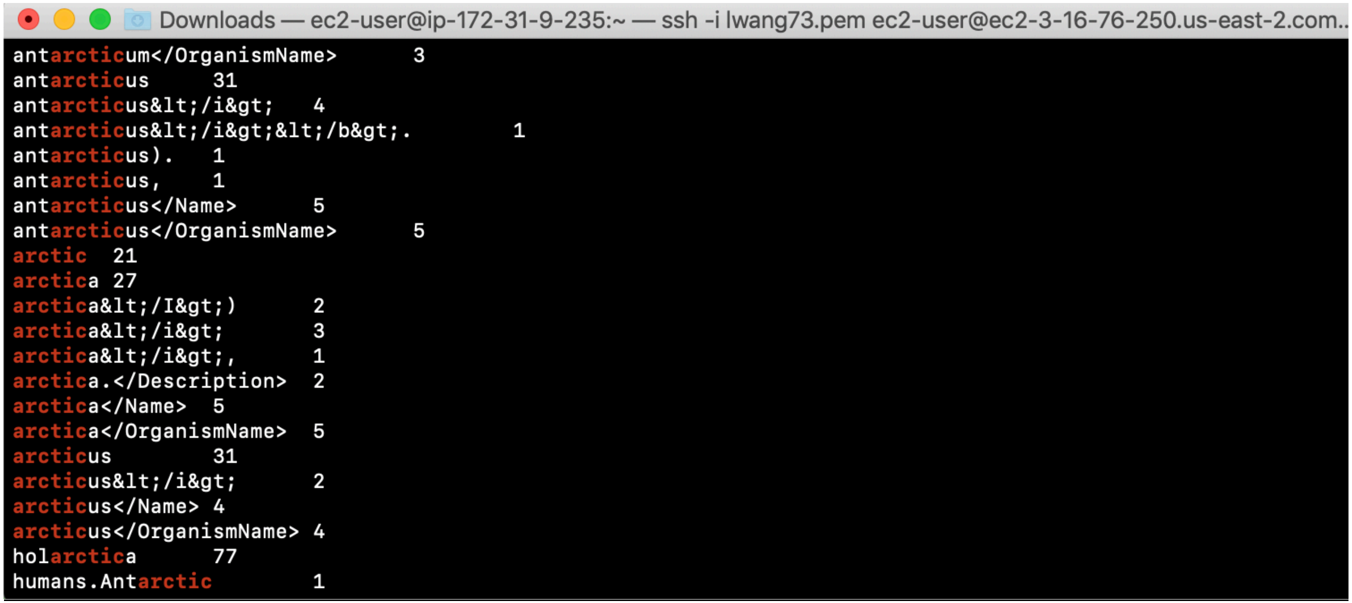
```
                Merged Map outputs=2
                GC time elapsed (ms)=569
                CPU time spent (ms)=39990
                Physical memory (bytes) snapshot=745447424
                Virtual memory (bytes) snapshot=6448369664
                Total committed heap usage (bytes)=526909440
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=231153099
        File Output Format Counters
                Bytes Written=20056175

real    0m39.222s
user    0m3.779s
sys     0m0.263s
[ec2-user@ip-172-31-9-235 ~]$ 
```

```
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -du /data/wordcount1/
0          /data/wordcount1/_SUCCESS
20056175   /data/wordcount1/part-r-00000
```

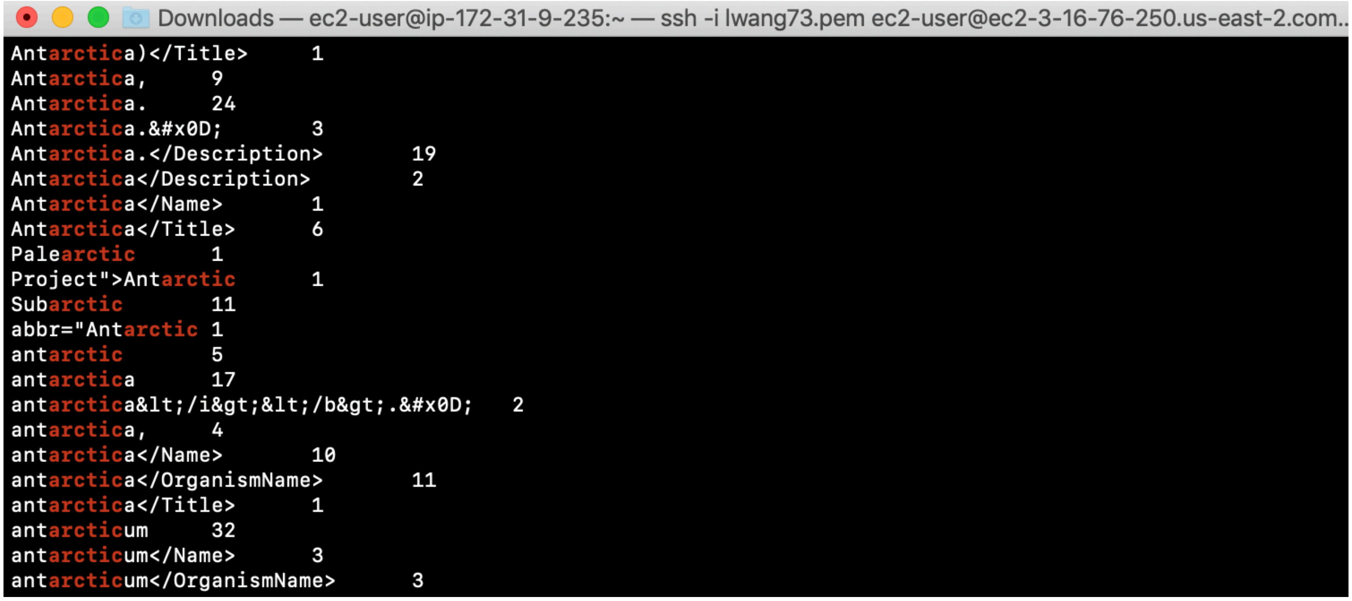

Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com.

```
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -cat /data/wordcount1/part-r-00000 | grep arctic
&lt;I&gt;holarctica&lt;/I&gt;	28
&lt;I&gt;holarctica&lt;/I&gt;&lt;/B&gt;.	8
&lt;I&gt;holarctica&lt;/I&gt;,	1
&lt;I&gt;palearctica&lt;/I&gt;	4
&lt;i&gt;holarctica&lt;/i&gt;	1
(Antarctic	3
(Antarctica)	1
(Antarctica),	11
<Label>Antarctic	1
<Name>Antarctic	3
<Name>Antarctica	1
<Strain>Antarctic	1
<Title>Antarctic	5
Antarctic	137
Antarctic,	1
Antarctic.	2
Antarctic.</Description>	1
Antarctic.</Title>	1
Antarctic</Title>	4
Antarctica	16
Antarctica)</Title>	1
```

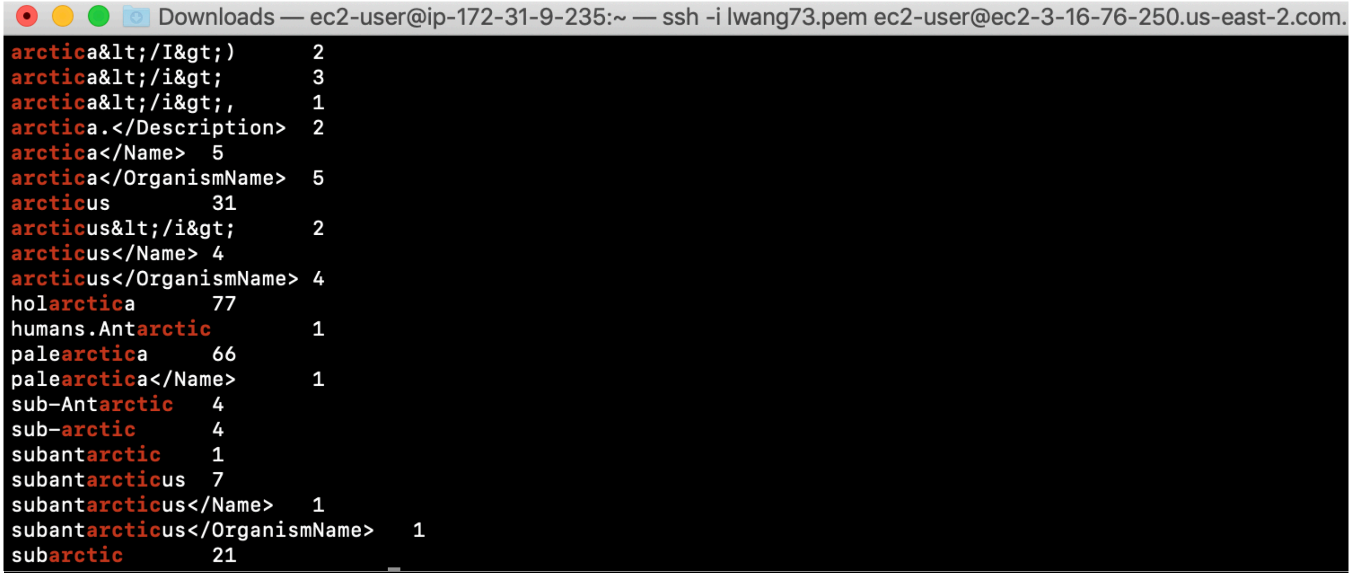
Lavinia Wang #1473704


Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

```
antarcticum</OrganismName>      3
antarcticus     31
antarcticus&lt;/i&gt;   4
antarcticus&lt;/i&gt;&lt;/b&gt;.        1
antarcticus).   1
antarcticus,    1
antarcticus</Name>      5
antarcticus</OrganismName>      5
arctic  21
arctica 27
arctica&lt;/I&gt;)      2
arctica&lt;/i&gt;       3
arctica&lt;/i&gt;,      1
arctica.</Description>  2
arctica</Name>  5
arctica</OrganismName>  5
arcticus        31
arcticus&lt;/i&gt;      2
arcticus</Name> 4
arcticus</OrganismName> 4
holarctica      77
humans.Antarctic        1
```


Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com..

```
Antarctica)</Title>     1
Antarctica,     9
Antarctica.     24
Antarctica.&#x0D;       3
Antarctica.</Description>       19
Antarctica</Description>        2
Antarctica</Name>       1
Antarctica</Title>      6
Palearctic      1
Project">Antarctic      1
Subarctic       11
abbr="Antarctic 1
antarctic       5
antarctica      17
antarctica&lt;/i&gt;&lt;/b&gt;.&#x0D;   2
antarctica,     4
antarctica</Name>       10
antarctica</OrganismName>       11
antarctica</Title>      1
antarcticum     32
antarcticum</Name>      3
antarcticum</OrganismName>      3
```


Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-3-16-76-250.us-east-2.com.

```
arctica&lt;/I&gt;)        2
arctica&lt;/i&gt;         3
arctica&lt;/i&gt;,        1
arctica.</Description>  2
arctica</Name>  5
arctica</OrganismName>  5
arcticus        31
arcticus&lt;/i&gt;        2
arcticus</Name> 4
arcticus</OrganismName> 4
holarctica      77
humans.Antarctic        1
palearctica     66
palearctica</Name>      1
sub-Antarctic   4
sub-arctic      4
subantarctic    1
subantarcticus  7
subantarcticus</Name>   1
subantarcticus</OrganismName>   1
subarctic       21
```

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

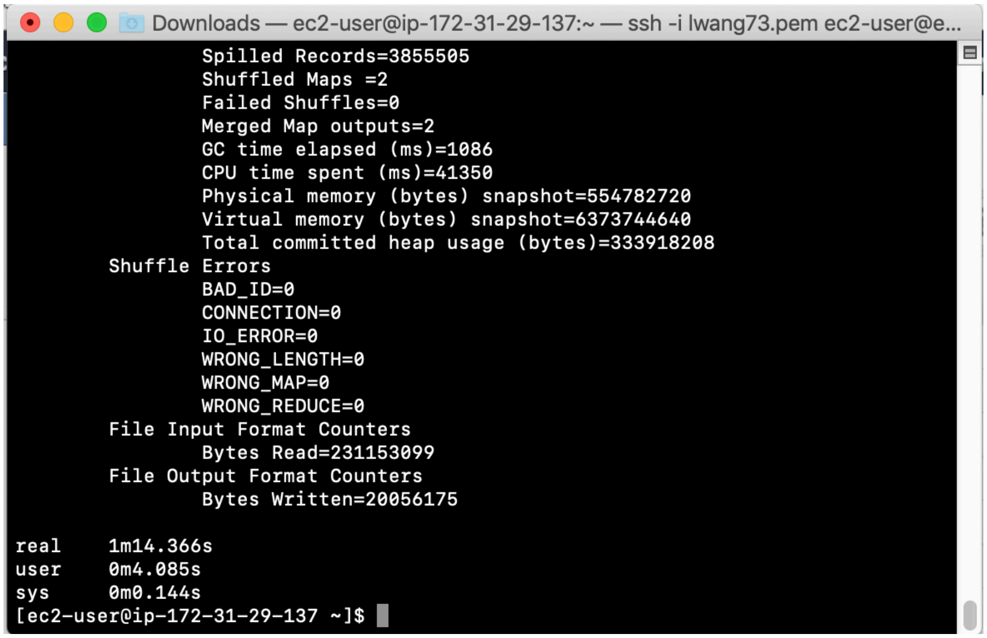
The single-node Hadoop instance from assignment 1 ran the word count job in 1 minute 14.366 seconds. The three-node instance ran the job 39.222 seconds which is roughly twice as fast. The speed increase is due to the additional nodes, however, I would have expected a more than doubling of speed, based just on the number of nodes. It's likely that network speed and block distribution have impacted speed.

Running the following command shows that the file was only split into two blocks and where they are located.

```
[ec2-user@ip-172-31-9-235 ~]$ hdfs fsck /data/bioproject.xml -files -blocks
-locations
Connecting to namenode via http://ip-172-31-9-235.us-east-
2.compute.internal:50070
FSCK started by ec2-user (auth:SIMPLE) from /172.31.9.235 for path
/data/bioproject.xml at Mon May 20 05:26:12 UTC 2019
/data/bioproject.xml 231149003 bytes, 2 block(s):  OK
0. BP-177962531-172.31.9.235-1558313129568:blk_1073741826_1002 len=134217728
repl=2 [172.31.7.186:50010, 172.31.6.219:50010]
1. BP-177962531-172.31.9.235-1558313129568:blk_1073741827_1003 len=96931275
repl=2 [172.31.7.186:50010, 172.31.6.219:50010]

Status: HEALTHY
 Total size: 231149003 B
 Total dirs: 0
 Total files:    1
 Total symlinks:     0
 Total blocks (validated):    2 (avg. block size 115574501 B)
 Minimally replicated blocks: 2 (100.0 %)
 Over-replicated blocks:  0 (0.0 %)
 Under-replicated blocks: 0 (0.0 %)
 Mis-replicated blocks:       0 (0.0 %)
 Default replication factor:  2
 Average block replication:   2.0
 Corrupt blocks:      0
 Missing replicas:        0 (0.0 %)
 Number of data-nodes:        3
 Number of racks:         1
FSCK ended at Mon May 20 05:26:12 UTC 2019 in 2 milliseconds


The filesystem under path '/data/bioproject.xml' is HEALTHY
```

This would explain the half-time only gain in speed.

Here are the times from assignment 1:



Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@e...

```
                Spilled Records=3855505
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=1086
                CPU time spent (ms)=41350
                Physical memory (bytes) snapshot=554782720
                Virtual memory (bytes) snapshot=6373744640
                Total committed heap usage (bytes)=333918208
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=231153099
        File Output Format Counters
                Bytes Written=20056175

real    1m14.366s
user    0m4.085s
sys     0m0.144s
[ec2-user@ip-172-31-29-137 ~]$ 
```

Part 2: Hive

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

Create and load tables:

```
CREATE TABLE lineorder (
  lo_orderkey        INT,
  lo_linenumber       INT,
  lo_custkey         INT,
  lo_partkey         INT,
  lo_suppkey          INT,
  lo_orderdate        INT,
  lo_orderpriority    VARCHAR(15),
  lo_shippriority     VARCHAR(1),
  lo_quantity         INT,
  lo_extendedprice    INT,
  lo_ordertotalprice  INT,
  lo_discount         INT,
  lo_revenue          INT,
  lo_supplycost       INT,
  lo_tax              INT,
  lo_commitdate       INT,
  lo_shipmode         VARCHAR(10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

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

CREATE TABLE dwdate (
  d_datekey           INT,
  d_date              VARCHAR(19),
  d_dayofweek         VARCHAR(10),
  d_month             VARCHAR(10),
  d_year              INT,
  d_yearmonthnum      INT,
  d_yearmonth         VARCHAR(8),
  d_daynuminweek      INT,
  d_daynuminmonth     INT,
  d_daynuminyear      INT,
  d_monthnuminyear    INT,
  d_weeknuminyear     INT,
  d_sellingseason     VARCHAR(13),
  d_lastdayinweekfl   VARCHAR(1),
  d_lastdayinmonthfl  VARCHAR(1),
  d_holidayfl         VARCHAR(1),
  d_weekdayfl         VARCHAR(1)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

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

CREATE TABLE part (
  p_partkey     INT,
  p_name        VARCHAR(22),
  p_mfgr        VARCHAR(6),
  p_category    VARCHAR(7),
  p_brand1      VARCHAR(9),
  p_color       VARCHAR(11),
  p_type        VARCHAR(25),
  p_size        INT,
  p_container   VARCHAR(10)
)ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

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

CREATE TABLE supplier (
  s_suppkey     INT,
  s_name        VARCHAR(25),
  s_address     VARCHAR(25),
  s_city        VARCHAR(10),
  s_nation      VARCHAR(15),
  s_region      VARCHAR(12),
  s_phone       VARCHAR(15)
)ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

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

1.2 Time to execute = 36.395 seconds

```
hive> 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;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20190521013853_cd4c2846-571d-49bd-9ed6-0da203aacb0e
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-
2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at: /tmp/ec2-user/ec2-user_20190521013853_cd4c2846-571d-49bd-9ed6-
0da203aacb0e.log
2019-05-21 01:38:59     Starting to launch local task to process map join;  maximum memory =
477626368
```

```
2019-05-21 01:39:00     Dump the side-table for tag: 1 with group count: 31 into file: file:/tmp/ec2-
user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-38-53_599_1551666118767653938-
1/-local-10005/HashTable-Stage-2/MapJoin-mapfile01--.hashtable
2019-05-21 01:39:00     Uploaded 1 File to: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-
3f8ad466871e/hive_2019-05-21_01-38-53_599_1551666118767653938-1/-local-10005/HashTable-Stage-
2/MapJoin-mapfile01--.hashtable (945 bytes)
2019-05-21 01:39:00     End of local task; Time Taken: 1.232 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1558399145275_0001, Tracking URL = http://ip-172-31-9-235.us-east-
2.compute.internal:8088/proxy/application_1558399145275_0001/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0001
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2019-05-21 01:39:08,201 Stage-2 map = 0%,  reduce = 0%
2019-05-21 01:39:21,323 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 9.7 sec
2019-05-21 01:39:24,590 Stage-2 map = 56%,  reduce = 0%, Cumulative CPU 13.02 sec
2019-05-21 01:39:25,749 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 13.72 sec
2019-05-21 01:39:28,912 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 15.22 sec
MapReduce Total cumulative CPU time: 15 seconds 220 msec
Ended Job = job_1558399145275_0001
MapReduce Jobs Launched:
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 15.22 sec   HDFS Read: 594368438 HDFS Write: 12
SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 220 msec
OK
14215822897
Time taken: 36.395 seconds, Fetched: 1 row(s)
```

1.3 Time to execute = 31.263 seconds

```
hive> 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;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20190521014210_0b745812-8c9a-4c6d-b2b7-8dedb2fd9854
Total jobs = 1
```

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-
2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at: /tmp/ec2-user/ec2-user_20190521014210_0b745812-8c9a-4c6d-b2b7-
8dedb2fd9854.log
2019-05-21 01:42:14	Starting to launch local task to process map join;	maximum memory =
477626368
2019-05-21 01:42:15	Dump the side-table for tag: 1 with group count: 7 into file: file:/tmp/ec2-
user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-42-10_401_7461872208873411953-
1/-local-10005/HashTable-Stage-2/MapJoin-mapfile11--.hashtable
2019-05-21 01:42:16	Uploaded 1 File to: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-
3f8ad466871e/hive_2019-05-21_01-42-10_401_7461872208873411953-1/-local-10005/HashTable-Stage-
2/MapJoin-mapfile11--.hashtable (414 bytes)
2019-05-21 01:42:16	End of local task; Time Taken: 1.163 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1558399145275_0002, Tracking URL = http://ip-172-31-9-235.us-east-
2.compute.internal:8088/proxy/application_1558399145275_0002/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0002
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2019-05-21 01:42:21,508 Stage-2 map = 0%,  reduce = 0%
2019-05-21 01:42:33,313 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 3.05 sec
2019-05-21 01:42:34,425 Stage-2 map = 56%,  reduce = 0%, Cumulative CPU 6.8 sec
2019-05-21 01:42:35,470 Stage-2 map = 78%,  reduce = 0%, Cumulative CPU 11.74 sec
2019-05-21 01:42:36,496 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 12.86 sec
2019-05-21 01:42:40,605 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 14.48 sec
MapReduce Total cumulative CPU time: 14 seconds 480 msec
Ended Job = job_1558399145275_0002
MapReduce Jobs Launched:
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 14.48 sec   HDFS Read: 594368557 HDFS Write: 11
SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 480 msec
OK
4435791464
Time taken: 31.263 seconds, Fetched: 1 row(s)
```

Lavinia Wang #1473704

2.1 Time to execute = 114.283 seconds

```
hive> 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;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20190521014433_b33e577a-c400-4c15-9266-3bf7cdf04727
Total jobs = 6
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at: /tmp/ec2-user/ec2-user_20190521014433_b33e577a-c400-4c15-9266-3bf7cdf04727.log
2019-05-21 01:44:38     Starting to launch local task to process map join;  maximum memory = 477626368
2019-05-21 01:44:39     Dump the side-table for tag: 1 with group count: 2556 into file: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile51--.hashtable
2019-05-21 01:44:39     Uploaded 1 File to: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-1/-local-10014/HashTable-Stage-13/MapJoin-mapfile51--.hashtable (67039 bytes)
2019-05-21 01:44:39     End of local task; Time Taken: 1.183 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 6
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1558399145275_0003, Tracking URL = http://ip-172-31-9-235.us-east-2.compute.internal:8088/proxy/application_1558399145275_0003/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0003
Hadoop job information for Stage-13: number of mappers: 3; number of reducers: 0
2019-05-21 01:44:45,929 Stage-13 map = 0%,  reduce = 0%
2019-05-21 01:45:01,911 Stage-13 map = 33%,  reduce = 0%, Cumulative CPU 12.7 sec
2019-05-21 01:45:06,121 Stage-13 map = 50%,  reduce = 0%, Cumulative CPU 21.24 sec
2019-05-21 01:45:08,260 Stage-13 map = 67%,  reduce = 0%, Cumulative CPU 22.82 sec
2019-05-21 01:45:13,584 Stage-13 map = 83%,  reduce = 0%, Cumulative CPU 28.51 sec
2019-05-21 01:45:14,607 Stage-13 map = 100%,  reduce = 0%, Cumulative CPU 30.18 sec
MapReduce Total cumulative CPU time: 30 seconds 180 msec
Ended Job = job_1558399145275_0003
```

```
Stage-15 is selected by condition resolver.
Stage-16 is filtered out by condition resolver.
Stage-2 is filtered out by condition resolver.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-
2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at: /tmp/ec2-user/ec2-user_20190521014433_b33e577a-c400-4c15-9266-3bf7cdf04727.log
2019-05-21 01:45:20	Starting to launch local task to process map join;	maximum memory = 
477626368
2019-05-21 01:45:21	Dump the side-table for tag: 1 with group count: 7883 into file: file:/tmp/ec2-
user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-
1/-local-10010/HashTable-Stage-10/MapJoin-mapfile31--.hashtable
2019-05-21 01:45:21	Uploaded 1 File to: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-
3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-1/-local-10010/HashTable-Stage-
10/MapJoin-mapfile31--.hashtable (249337 bytes)
2019-05-21 01:45:21	End of local task; Time Taken: 1.761 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3 out of 6
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1558399145275_0004, Tracking URL = http://ip-172-31-9-235.us-east-
2.compute.internal:8088/proxy/application_1558399145275_0004/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0004
Hadoop job information for Stage-10: number of mappers: 1; number of reducers: 0
2019-05-21 01:45:27,071 Stage-10 map = 0%,  reduce = 0%
2019-05-21 01:45:40,419 Stage-10 map = 45%,  reduce = 0%, Cumulative CPU 10.64 sec
2019-05-21 01:45:44,530 Stage-10 map = 100%,  reduce = 0%, Cumulative CPU 14.75 sec
MapReduce Total cumulative CPU time: 14 seconds 750 msec
Ended Job = job_1558399145275_0004
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-
2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Execution log at: /tmp/ec2-user/ec2-user_20190521014433_b33e577a-c400-4c15-9266-3bf7cdf04727.log
2019-05-21 01:45:49	Starting to launch local task to process map join;	maximum memory = 
477626368
2019-05-21 01:45:51	Dump the side-table for tag: 1 with group count: 378 into file: file:/tmp/ec2-
user/5a9d12ba-e847-4d98-9d5f-3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-
1/-local-10008/HashTable-Stage-4/MapJoin-mapfile21--.hashtable
```

```
2019-05-21 01:45:51	Uploaded 1 File to: file:/tmp/ec2-user/5a9d12ba-e847-4d98-9d5f-
3f8ad466871e/hive_2019-05-21_01-44-33_667_5626930748980129596-1/-local-10008/HashTable-Stage-
4/MapJoin-mapfile21--.hashtable (7792 bytes)
2019-05-21 01:45:51	End of local task; Time Taken: 1.312 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 4 out of 6
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1558399145275_0005, Tracking URL = http://ip-172-31-9-235.us-east-
2.compute.internal:8088/proxy/application_1558399145275_0005/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0005
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2019-05-21 01:45:56,632 Stage-4 map = 0%,  reduce = 0%
2019-05-21 01:46:02,806 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 3.56 sec
2019-05-21 01:46:08,981 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 5.2 sec
MapReduce Total cumulative CPU time: 5 seconds 200 msec
Ended Job = job_1558399145275_0005
Launching Job 5 out of 6
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1558399145275_0006, Tracking URL = http://ip-172-31-9-235.us-east-
2.compute.internal:8088/proxy/application_1558399145275_0006/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1558399145275_0006
Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1
2019-05-21 01:46:16,561 Stage-5 map = 0%,  reduce = 0%
2019-05-21 01:46:21,761 Stage-5 map = 100%,  reduce = 0%, Cumulative CPU 1.02 sec
2019-05-21 01:46:26,896 Stage-5 map = 100%,  reduce = 100%, Cumulative CPU 2.3 sec
MapReduce Total cumulative CPU time: 2 seconds 300 msec
Ended Job = job_1558399145275_0006
MapReduce Jobs Launched: 
Stage-Stage-13: Map: 3   Cumulative CPU: 30.18 sec   HDFS Read: 594359050 HDFS Write: 184733291 
SUCCESS
Stage-Stage-10: Map: 1   Cumulative CPU: 14.75 sec   HDFS Read: 184739547 HDFS Write: 8750735 
SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 5.2 sec   HDFS Read: 8762980 HDFS Write: 9913 
SUCCESS
```

```
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 2.3 sec   HDFS Read: 15707 HDFS Write: 6937
SUCCESS
Total MapReduce CPU Time Spent: 52 seconds 430 msec
OK
567838207  1992  MFGR#121
610663790  1992  MFGR#1210
550769662  1992  MFGR#1211
649205856  1992  MFGR#1212
624031241  1992  MFGR#1213
670488468  1992  MFGR#1214
633152470  1992  MFGR#1215
674846781  1992  MFGR#1216
675093435  1992  MFGR#1217
600202070  1992  MFGR#1218
538043594  1992  MFGR#1219
655326672  1992  MFGR#122
540262882  1992  MFGR#1220
556120633  1992  MFGR#1221
590762777  1992  MFGR#1222
535448651  1992  MFGR#1223
703752611  1992  MFGR#1224
570832868  1992  MFGR#1225
614061593  1992  MFGR#1226
581759388  1992  MFGR#1227
644642592  1992  MFGR#1228
640858430  1992  MFGR#1229
789755835  1992  MFGR#123
468535087  1992  MFGR#1230
592436656  1992  MFGR#1231
664275152  1992  MFGR#1232
613885100  1992  MFGR#1233
667399281  1992  MFGR#1234
640290070  1992  MFGR#1235
501892561  1992  MFGR#1236
591481503  1992  MFGR#1237
477423770  1992  MFGR#1238
638259374  1992  MFGR#1239
572354196  1992  MFGR#124
740479248  1992  MFGR#1240
478777095  1992  MFGR#125
592174616  1992  MFGR#126
706151632  1992  MFGR#127
542306646  1992  MFGR#128
581987352  1992  MFGR#129
823087702  1993  MFGR#121
648160706  1993  MFGR#1210
634743898  1993  MFGR#1211
785639283  1993  MFGR#1212
```

638255029	1993	MFGR#1213
616837237	1993	MFGR#1214
634687975	1993	MFGR#1215
638353900	1993	MFGR#1216
663372951	1993	MFGR#1217
683985855	1993	MFGR#1218
646950033	1993	MFGR#1219
622532984	1993	MFGR#122
530830127	1993	MFGR#1220
543346337	1993	MFGR#1221
756921203	1993	MFGR#1222
533544350	1993	MFGR#1223
915916085	1993	MFGR#1224
473007381	1993	MFGR#1225
739036124	1993	MFGR#1226
592178887	1993	MFGR#1227
583507058	1993	MFGR#1228
617453491	1993	MFGR#1229
637863868	1993	MFGR#123
625534310	1993	MFGR#1230
580327635	1993	MFGR#1231
697373098	1993	MFGR#1232
515571416	1993	MFGR#1233
651935758	1993	MFGR#1234
575779480	1993	MFGR#1235
591878667	1993	MFGR#1236
609618576	1993	MFGR#1237
444614010	1993	MFGR#1238
595256327	1993	MFGR#1239
660586237	1993	MFGR#124
788730059	1993	MFGR#1240
616224539	1993	MFGR#125
617126754	1993	MFGR#126
654438324	1993	MFGR#127
731657001	1993	MFGR#128
548048395	1993	MFGR#129
564405648	1994	MFGR#121
645404849	1994	MFGR#1210
631620635	1994	MFGR#1211
568332348	1994	MFGR#1212
678785857	1994	MFGR#1213
534002330	1994	MFGR#1214
654400242	1994	MFGR#1215
558646341	1994	MFGR#1216
687845641	1994	MFGR#1217
546674347	1994	MFGR#1218
567272942	1994	MFGR#1219
659884062	1994	MFGR#122

562582172	1994	MFGR#1220
598618997	1994	MFGR#1221
601016441	1994	MFGR#1222
555134404	1994	MFGR#1223
737422302	1994	MFGR#1224
570745955	1994	MFGR#1225
746302245	1994	MFGR#1226
651707481	1994	MFGR#1227
573693547	1994	MFGR#1228
647918373	1994	MFGR#1229
580449592	1994	MFGR#123
493270412	1994	MFGR#1230
603546148	1994	MFGR#1231
719865331	1994	MFGR#1232
638982238	1994	MFGR#1233
743247677	1994	MFGR#1234
598680959	1994	MFGR#1235
615726097	1994	MFGR#1236
542569815	1994	MFGR#1237
573510781	1994	MFGR#1238
579855853	1994	MFGR#1239
684573322	1994	MFGR#124
873735737	1994	MFGR#1240
560488304	1994	MFGR#125
657036514	1994	MFGR#126
622571183	1994	MFGR#127
586845664	1994	MFGR#128
534541525	1994	MFGR#129
706469511	1995	MFGR#121
602892803	1995	MFGR#1210
645166092	1995	MFGR#1211
613289283	1995	MFGR#1212
599586479	1995	MFGR#1213
562570804	1995	MFGR#1214
672528755	1995	MFGR#1215
669000972	1995	MFGR#1216
725362449	1995	MFGR#1217
657026635	1995	MFGR#1218
519659003	1995	MFGR#1219
724727741	1995	MFGR#122
517956131	1995	MFGR#1220
635741351	1995	MFGR#1221
564368410	1995	MFGR#1222
600665149	1995	MFGR#1223
762700351	1995	MFGR#1224
671669586	1995	MFGR#1225
572568748	1995	MFGR#1226
530361300	1995	MFGR#1227

633357085	1995	MFGR#1228
547960244	1995	MFGR#1229
660711077	1995	MFGR#123
602735858	1995	MFGR#1230
499852146	1995	MFGR#1231
715300753	1995	MFGR#1232
557149571	1995	MFGR#1233
710023059	1995	MFGR#1234
622425239	1995	MFGR#1235
634565501	1995	MFGR#1236
572847270	1995	MFGR#1237
549318912	1995	MFGR#1238
593851712	1995	MFGR#1239
585421815	1995	MFGR#124
707207888	1995	MFGR#1240
538246872	1995	MFGR#125
605799021	1995	MFGR#126
665978112	1995	MFGR#127
646960956	1995	MFGR#128
508749401	1995	MFGR#129
523879145	1996	MFGR#121
643645053	1996	MFGR#1210
595065339	1996	MFGR#1211
674626440	1996	MFGR#1212
496297087	1996	MFGR#1213
583249505	1996	MFGR#1214
702184857	1996	MFGR#1215
601809334	1996	MFGR#1216
704898387	1996	MFGR#1217
528843086	1996	MFGR#1218
586246330	1996	MFGR#1219
712110492	1996	MFGR#122
518444215	1996	MFGR#1220
499319414	1996	MFGR#1221
679469356	1996	MFGR#1222
628762754	1996	MFGR#1223
724844856	1996	MFGR#1224
660620587	1996	MFGR#1225
667674729	1996	MFGR#1226
483838085	1996	MFGR#1227
609855391	1996	MFGR#1228
658959557	1996	MFGR#1229
566217852	1996	MFGR#123
528879998	1996	MFGR#1230
589481194	1996	MFGR#1231
702805896	1996	MFGR#1232
663679947	1996	MFGR#1233
571149450	1996	MFGR#1234

478648074	1996	MFGR#1235
568249365	1996	MFGR#1236
592616167	1996	MFGR#1237
466676148	1996	MFGR#1238
670693719	1996	MFGR#1239
560667719	1996	MFGR#124
821167950	1996	MFGR#1240
476864333	1996	MFGR#125
558030884	1996	MFGR#126
635873891	1996	MFGR#127
551010618	1996	MFGR#128
560570630	1996	MFGR#129
587013207	1997	MFGR#121
616287892	1997	MFGR#1210
548588761	1997	MFGR#1211
589593892	1997	MFGR#1212
424306670	1997	MFGR#1213
511971910	1997	MFGR#1214
631772246	1997	MFGR#1215
692135140	1997	MFGR#1216
777994957	1997	MFGR#1217
707053720	1997	MFGR#1218
561169527	1997	MFGR#1219
664916245	1997	MFGR#122
594466157	1997	MFGR#1220
588848171	1997	MFGR#1221
528988960	1997	MFGR#1222
537098211	1997	MFGR#1223
674763166	1997	MFGR#1224
450402292	1997	MFGR#1225
701360722	1997	MFGR#1226
506011570	1997	MFGR#1227
585578737	1997	MFGR#1228
622744016	1997	MFGR#1229
646503168	1997	MFGR#123
571800941	1997	MFGR#1230
502601790	1997	MFGR#1231
677924656	1997	MFGR#1232
534455976	1997	MFGR#1233
714934715	1997	MFGR#1234
767151420	1997	MFGR#1235
618877179	1997	MFGR#1236
639638057	1997	MFGR#1237
401953419	1997	MFGR#1238
610756714	1997	MFGR#1239
543248087	1997	MFGR#124
675132692	1997	MFGR#1240
479099365	1997	MFGR#125

```
570696568  1997  MFGR#126
583074592  1997  MFGR#127
695133104  1997  MFGR#128
655638776  1997  MFGR#129
344575925  1998  MFGR#121
417152416  1998  MFGR#1210
317068168  1998  MFGR#1211
374341516  1998  MFGR#1212
332740903  1998  MFGR#1213
304873002  1998  MFGR#1214
366101132  1998  MFGR#1215
379133898  1998  MFGR#1216
359508497  1998  MFGR#1217
320623334  1998  MFGR#1218
346182862  1998  MFGR#1219
312440027  1998  MFGR#122
348123961  1998  MFGR#1220
339845398  1998  MFGR#1221
355416161  1998  MFGR#1222
344889822  1998  MFGR#1223
396906691  1998  MFGR#1224
290208878  1998  MFGR#1225
419415707  1998  MFGR#1226
358466340  1998  MFGR#1227
251549955  1998  MFGR#1228
383138860  1998  MFGR#1229
296330561  1998  MFGR#123
437181243  1998  MFGR#1230
398944492  1998  MFGR#1231
424062455  1998  MFGR#1232
406967188  1998  MFGR#1233
428867240  1998  MFGR#1234
352277781  1998  MFGR#1235
361827086  1998  MFGR#1236
341618569  1998  MFGR#1237
244739231  1998  MFGR#1238
414151803  1998  MFGR#1239
330082371  1998  MFGR#124
415312453  1998  MFGR#1240
360289624  1998  MFGR#125
341657580  1998  MFGR#126
377507061  1998  MFGR#127
361416497  1998  MFGR#128
318769573  1998  MFGR#129
Time taken: 114.283 seconds, Fetched: 280 row(s)
```

Perform the following transform operation using SELECT TRANSFORM on the customer table by creating a new table. Your new target table should have only three columns, c_custkey (no changes), c_address, and c_city.

For the c_address column, shorten it to 8 characters (i.e., if the value is longer, remove extra characters, but otherwise keep it as-is). For c_city, add a space and a # to indicate the digit at the end (e.g., UNITED KI2 => UNITED KI #2, or INDONESIA4 => INDONESIA #4). Make sure to modify the columns of the target table accordingly (since you are introducing longer columns).

customer.py

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

for line in sys.stdin:
    line = line.strip().split(',')
    line[2] = line[2][:8]
    line[3] = line[3][:len(line[3])-1] + ' #' + line[3][len(line[3])-1]
    print '\t'.join(line)
```

create and load customer table

```
CREATE TABLE customer (
  c_custkey     INT,
  c_name        VARCHAR(25),
  c_address     VARCHAR (25),
  c_city        VARCHAR (10),
  c_nation      VARCHAR (15),
  c_region      VARCHAR (12),
  c_phone       VARCHAR (15),
  c_mktsegment  VARCHAR (10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;
```

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

Create customer2 table

```
create table customer2 (
  c_custkey     INT,
  c_name        VARCHAR(25),
  c_address     VARCHAR (8),
  c_city        VARCHAR (12),
  c_nation      VARCHAR (15),
  c_region      VARCHAR (12),
  c_phone       VARCHAR (15),
  c_mktsegment  VARCHAR (10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' STORED AS TEXTFILE;
```

Select transform

INSERT OVERWRITE TABLE customer2 SELECT TRANSFORM (c_custkey, c_name, c_address, c_city, c_nation, c_region, c_phone, c_mktsegment) USING 'python customer.py' AS (c_custkey, c_name, c_address, c_city, c_nation, c_region, c_phone, c_mktsegment) FROM customer;

test query:

```
hive> select * from customer2 limit 5;

OK

1      Customer#000000001     j5JsirBM       MOROCCO   #0    MOROCCO AFRICA  25-989-741-2988 BUILDING

2      Customer#000000002     487LW1do       JORDAN    #1    JORDAN  MIDDLE EAST     23-768-687-3665  AUTOMOBILE

3      Customer#000000003     fkRGN8n ARGENTINA #7    ARGENTINA      AMERICA 11-719-748-3364 AUTOMOBILE

4      Customer#000000004     4u58h f EGYPT     #4    EGYPT   MIDDLE EAST     14-128-190-5944 MACHINERY

5      Customer#000000005     hwBtxkoB       CANADA    #5    CANADA  AMERICA 13-750-942-6364 HOUSEHOLD

Time taken: 0.051 seconds, Fetched: 5 row(s)
```

Part 3: Pig

Convert and load the data into Pig, implementing only queries 0.1, 0.2, 0.3. Do not implement all queries.

Check disk storage space in HDFS, 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).

0.1

```
lineorder = LOAD '/user/ec2-user/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
);

grouped = group lineorder all;
avg = FOREACH grouped GENERATE AVG(lineorder.lo_revenue);
DUMP avg;
```

Answer = 3634300.709514323

Execution time of 3 minutes, 34 seconds, 11 millisecond

0.2

```
grouped = group lineorder by lo_discount;
counted = FOREACH grouped GENERATE group as lo_discount, COUNT(lineorder.lo_extendedprice) as
cnt;
DUMP counted;
```

```
2019-05-21 05:20:36,325 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(0,544886)
(1,545834)
(2,546173)
(3,545293)
(4,545545)
(5,546395)
(6,544970)
(7,546192)
(8,544803)
(9,545309)
(10,545815)
2019-05-21 05:20:36,402 [main] INFO  org.apache.pig.Main - Pig script completed in 3 minutes, 47
seconds and 31 milliseconds (227031ms)
```

0.3

```
filtered = FILTER lineorder BY lo_discount < 3;
grouped = GROUP filtered BY lo_quantity;
summed = FOREACH grouped GENERATE group as lo_quantity, SUM(filtered.lo_revenue) as rev;
DUMP summed;
```

```
2019-05-21 06:04:22,932 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
- Total input paths to process : 1
(1,4879019020)
```

Lavinia Wang #1473704

```
(2,9644127315)
(3,14575887127)
(4,19360189865)
(5,24073923574)
(6,29125189531)
(7,33982891466)
(8,38671565454)
(9,43381602619)
(10,48619780003)
(11,53159489411)
(12,58264291629)
(13,62920595763)
(14,67451818069)
(15,73414895616)
(16,78360133885)
(17,82320521791)
(18,86995495639)
(19,93016668045)
(20,97258062753)
(21,100684344044)
(22,107454001560)
(23,112984674102)
(24,116527702603)
(25,123160894092)
(26,126451771059)
(27,132113291310)
(28,135413154368)
(29,141357789043)
(30,145181046794)
(31,149937771539)
(32,157770330201)
(33,161774040572)
(34,164150363629)
(35,170173151151)
(36,175712858188)
(37,178733976488)
(38,186428562667)
(39,187696104837)
(40,196345645204)
(41,199250645070)
(42,204966410590)
(43,209016181876)
(44,213245636104)
(45,217565230742)
(46,223784510215)
(47,229077142619)
(48,234125822088)
(49,236641410613)
```

```
(50,243791122644)
2019-05-21 06:04:22,987 [main] INFO  org.apache.pig.Main - Pig script completed in 3 minutes, 17 seconds and 45 milliseconds (197045 ms)
```

Part 4: Hadoop Streaming

Implement query **0.3** using Hadoop streaming with python. You don't need to implement other queries.

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

Set up commands to move lineorder.tbl into Hadoop directory.

```
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -mkdir /user/ec2-user/ssbm
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -put lineorder.tbl ssbm
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -ls ssbm
Found 1 items
-rw-r--r--   2 ec2-user supergroup  594313001 2019-05-21 03:01 ssbm/lineorder.tbl
```

myMapper.0.3.py

```
#!/usr/bin/python

import sys

for line in sys.stdin:
    line = line.strip()
    vals = line.split("|")

    discount = int(vals[11])
    quantity = vals[8]
    revenue = vals[12]

    if discount < 3:
        print "%s\t%s" % (quantity, revenue)
```

myReducer.0.3.py

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

curr_id = None
curr_tot = 0
id = None
```

```
# The input comes from standard input (line by line)
for line in sys.stdin:
    line = line.strip()
    # parse the line and split it by '\t'
    ln = line.split('\t')
    # grab the key
    id = ln[0]
    # grab the value (int)
    val = int(ln[1])

    if curr_id == id:
            curr_tot += val
    else:
        if curr_id: # output the sum, single key completed
            print '%s\t%d' % (curr_id, curr_tot)

        curr_id = id
        curr_tot = val

# output the last key
if curr_id == id:
    print '%s\t%d' % (curr_id, curr_tot)
```

command

```
[ec2-user@ip-172-31-9-235~]$ hadoop jar hadoop-streaming-2.6.4.jar -input
/user/ec2-user/ssbm -output /data/output16 -mapper myMapper.0.3.py -reducer
myReducer.0.3.py -file myReducer.0.3.py -file myMapper.0.3.py
```

Results

```
[ec2-user@ip-172-31-9-235~]$ hadoop fs -cat /data/output16/part-00000
1       4879019020
10      48619780003
11      53159489411
12      58264291629
13      62920595763
14      67451818069
15      73414895616
16      78360133885
17      82320521791
18      86995495639
19      93016668045
2       9644127315
20      97258062753
21      100684344044
22      107454001560
23      112984674102
24      116527702603
25      123160894092
26      126451771059
27      132113291310
28      135413154368
29      141357789043
3       14575887127
30      145181046794
```

```
31      149937771539
32      157770330201
33      161774040572
34      164150363629
35      170173151151
36      175712858188
37      178733976488
38      186428562667
39      187696104837
4       19360189865
40      196345645204
41      199250645070
42      204966410590
43      209016181876
44      213245636104
45      217565230742
46      223784510215
47      229077142619
48      234125822088
49      236641410613
5       24073923574
50      243791122644
6       29125189531
7       33982891466
8       38671565454
9       43381602619
```

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.