Alex Teboul

CSC 555: Mining Big Data

Project, Phase 1 (due Sunday, February 23rd)

In this part of the project (which will also 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. The modified Hive-style schema is.
http://rasinsrv07.cstcis.cti.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://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 HiveQL).

Part 1: Multi-node cluster

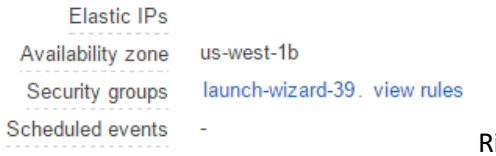
1) Your first step is to setup a multi-node cluster and re-run a simple wordcount. For this part, you will create a 3-node cluster (with a total of 1 master + 2 worker nodes). Include your master node in the "slaves" file, to make sure **all 3** nodes are working.
   You need to perform the following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

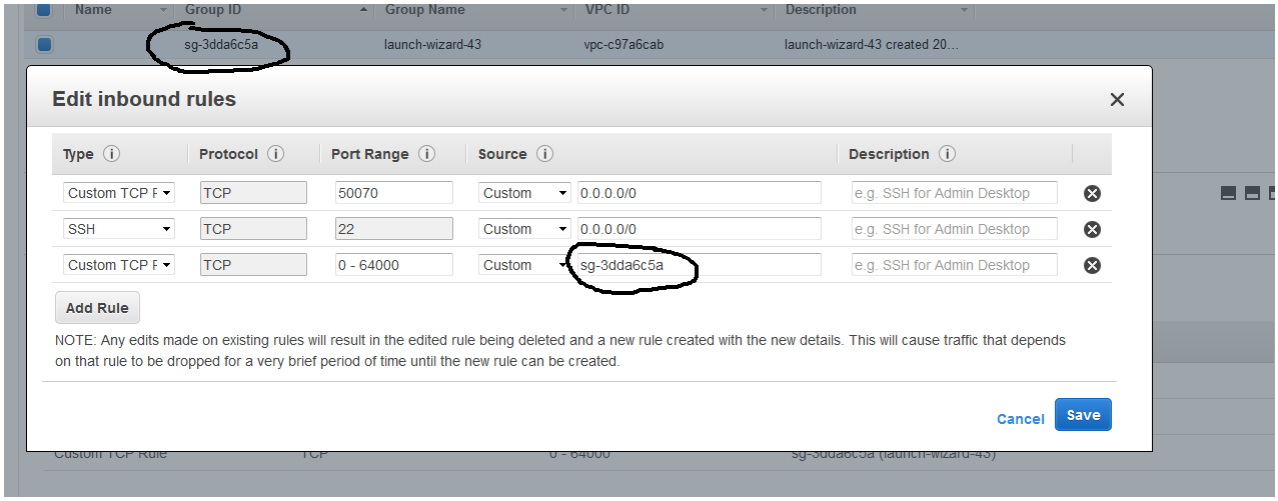
        b. Change your security group setting to open firewall access. We need to open the ports in two different ways. We will open port 50070 for the web interface in order to be able to see the cluster status in a browser. We will also set 0-64000 range opening up all ports. However, we will ensure that the 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).



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 please don't skip this step



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
```

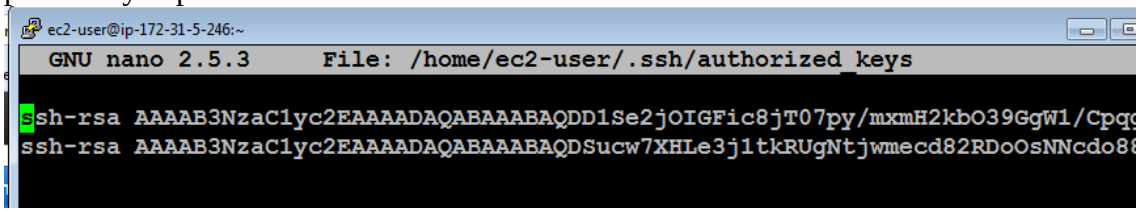
Finally, edit the slaves file and list your 3 nodes (master and 2 workers) using Private IPs

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

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.



   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).

1

```
[ec2-user@ip-172-31-38-169 ~]$ ssh 172.31.45.120
The authenticity of host '172.31.45.120 (172.31.45.120)' can't be established.
ECDSA key fingerprint is SHA256:JFt/Cu7VcwW+crTsThxob/UZdYtGZ0zzVujXmo0hwJg.
ECDSA key fingerprint is MD5:f1:7b:71:1d:2e:c9:0f:d0:a0:ab:73:51:28:c0:f7:d1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.45.120' (ECDSA) to the list of known hosts.
Last login: Sun Feb 23 23:40:02 2020 from 3.80.251.19

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-45-120 ~]$ exit
```

2

```
[ec2-user@ip-172-31-38-169 ~]$ ssh 172.31.34.17
The authenticity of host '172.31.34.17 (172.31.34.17)' can't be established.
ECDSA key fingerprint is SHA256:NSvtMF31zeT/iROxKSCq9iE7BNO05on4bBO/D8CBVIM.
ECDSA key fingerprint is MD5:78:13:6d:3b:bd:62:19:27:7e:46:27:a3:98:52:56:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.34.17' (ECDSA) to the list of known hosts.
Last login: Mon Feb 24 00:29:27 2020 from 67.167.219.77

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-34-17 ~]$
```

3

```
[ec2-user@ip-172-31-38-169 ~]$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:m5ftsNLpSIYipk+ltIKiZNcGL5gg4vDgvfSXcc/8DJM.
ECDSA key fingerprint is MD5:64:90:d6:2c:02:04:af:bc:3a:70:ef:35:8a:44:1c:aa.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Last login: Sun Feb 23 23:07:48 2020 from 67.167.219.77

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-38-169 ~]$
```

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)

```
[ec2-user@ip-172-31-38-169 ~]$ ls -al myHadoop.tar
-rw-rw-r-- 1 ec2-user ec2-user 314439680 Feb 24 01:05 myHadoop.tar
[ec2-user@ip-172-31-38-169 ~]$
```

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 <u>each</u> 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.

```
[ec2-user@ip-172-31-38-169 ~]$ scp myHadoop.tar ip-172-31-45-120:~/myHadoopWorker.tar
The authenticity of host 'ip-172-31-45-120 (172.31.45.120)' can't be established.
ECDSA key fingerprint is SHA256:JFt/Cu7VcwW+crTsThxob/UZdYtGZOzzVujXmo0hwJg.
ECDSA key fingerprint is MD5:f1:7b:71:1d:2e:c9:0f:d0:a0:ab:73:51:28:c0:f7:d1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ip-172-31-45-120' (ECDSA) to the list of known hosts.
myHadoop.tar                                  100%  300MB 120.7MB/s   00:02
[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-45-120 ~]$ ls -al
total 307100
drwx------ 3 ec2-user ec2-user      4096 Feb 24 01:24 .
drwxr-xr-x 3 root     root          4096 Feb 23 23:28 ..
-rw------- 1 ec2-user ec2-user         5 Feb 23 23:42 .bash_history
-rw-r--r-- 1 ec2-user ec2-user        18 Aug 30  2017 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user       193 Aug 30  2017 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user       124 Aug 30  2017 .bashrc
-rw-rw-r-- 1 ec2-user ec2-user 314439680 Feb 24 01:24 myHadoopWorker.tar
drwx------ 2 ec2-user ec2-user      4096 Feb 23 23:28 .ssh
[ec2-user@ip-172-31-45-120 ~]$
```

```
[ec2-user@ip-172-31-38-169 ~]$ scp myHadoop.tar ip-172-31-34-17:~/myHadoopWorker.tar
The authenticity of host 'ip-172-31-34-17 (172.31.34.17)' can't be established.
ECDSA key fingerprint is SHA256:NSvtMF31zeT/iROxKSCq9iE7BNO05on4bBO/D8CBVIM.
ECDSA key fingerprint is MD5:78:13:6d:3b:bd:62:19:27:7e:46:27:a3:98:52:56:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ip-172-31-34-17' (ECDSA) to the list of known hosts.
myHadoop.tar                                  100%  300MB 119.0MB/s   00:02
[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-34-17 ~]$ ls -al
total 307100
drwx------ 3 ec2-user ec2-user      4096 Feb 24 01:27 .
drwxr-xr-x 3 root     root          4096 Feb 23 23:29 ..
-rw------- 1 ec2-user ec2-user        35 Feb 24 00:46 .bash_history
-rw-r--r-- 1 ec2-user ec2-user        18 Aug 30  2017 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user       193 Aug 30  2017 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user       124 Aug 30  2017 .bashrc
-rw-rw-r-- 1 ec2-user ec2-user 314439680 Feb 24 01:27 myHadoopWorker.tar
drwx------ 2 ec2-user ec2-user      4096 Feb 24 00:43 .ssh
[ec2-user@ip-172-31-34-17 ~]$
```

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).

```
[ec2-user@ip-172-31-38-169 ~]$ jps
4510 Jps
3772 DataNode
3650 NameNode
3921 SecondaryNameNode
4158 NodeManager
4473 JobHistoryServer
4061 ResourceManager
```

```
-------------------------------------------------
Live datanodes (3):

Name: 172.31.34.17:50010 (ip-172-31-34-17.ec2.internal)
Hostname: ip-172-31-34-17.ec2.internal
Decommission Status : Normal
Configured Capacity: 8385892352 (7.81 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 1965101056 (1.83 GB)
DFS Remaining: 6420766720 (5.98 GB)
DFS Used%: 0.00%
DFS Remaining%: 76.57%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Feb 24 02:18:55 UTC 2020


Name: 172.31.38.169:50010 (ip-172-31-38-169.ec2.internal)
Hostname: ip-172-31-38-169.ec2.internal
Decommission Status : Normal
Configured Capacity: 31637610496 (29.46 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 2285305856 (2.13 GB)
DFS Remaining: 29352280064 (27.34 GB)
DFS Used%: 0.00%
DFS Remaining%: 92.78%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Feb 24 02:18:58 UTC 2020


Name: 172.31.45.120:50010 (ip-172-31-45-120.ec2.internal)
Hostname: ip-172-31-45-120.ec2.internal
Decommission Status : Normal
Configured Capacity: 8385892352 (7.81 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 1965162496 (1.83 GB)
DFS Remaining: 6420705280 (5.98 GB)
DFS Used%: 0.00%
DFS Remaining%: 76.57%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Feb 24 02:18:55 UTC 2020
```

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).

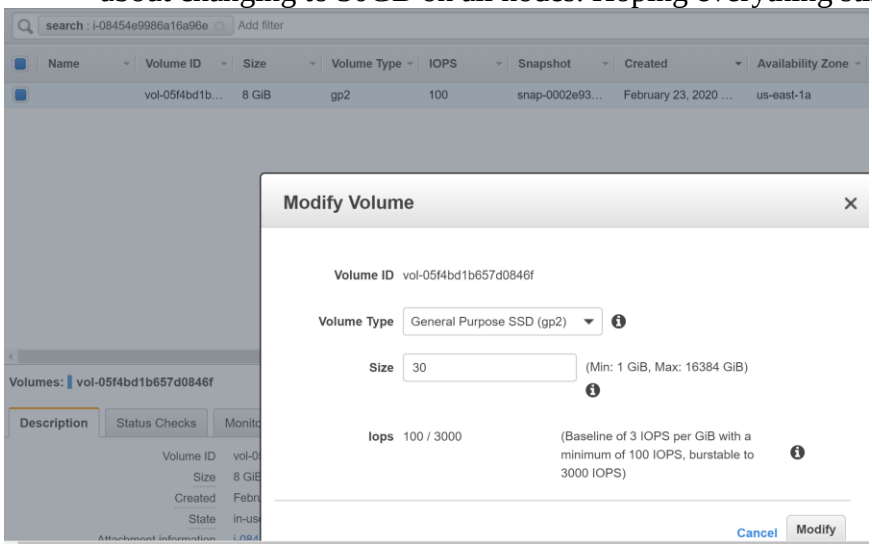
Submit a screenshot of your cluster status view.

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-45-120.ec2.internal (172.31.45.120:50010)	0	In Service	7.81 GB	24 KB	1.83 GB	5.98 GB	0	24 KB (0%)	0	2.6.4
ip-172-31-34-17.ec2.internal (172.31.34.17:50010)	0	In Service	7.81 GB	24 KB	1.83 GB	5.98 GB	0	24 KB (0%)	0	2.6.4
ip-172-31-38-169.ec2.internal (172.31.38.169:50010)	1	In Service	29.46 GB	24 KB	2.13 GB	27.34 GB	0	24 KB (0%)	0	2.6.4

- Followed the video setup instructions exactly, but capacity was not changed to 30G on the Workers in that video so I didn't here. But it seems to conflict with the 'Note' above about changing to 30GB on all nodes. Hoping everything still runs like this.



- Afraid of messing everything up by modifying the volume. Waiting on advice via the discussion forums on whether or not this works.

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

```
2020-02-24 03:08:02 (10.2 MB/s) - 'bioproject.xml' saved [231149003/231149003]

[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -put bioproject.xml /data/
[ec2-user@ip-172-31-38-169 ~]$ time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapredu
ce-examples-2.6.4.jar  wordcount /data/bioproject.xml /data/wordcount1
20/02/24 03:09:30 INFO client.RMProxy: Connecting to ResourceManager at /172.31.38.169:8032
20/02/24 03:09:31 INFO input.FileInputFormat: Total input paths to process : 1
20/02/24 03:09:31 INFO mapreduce.JobSubmitter: number of splits:2
20/02/24 03:09:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1582510523493_0001
20/02/24 03:09:31 INFO impl.YarnClientImpl: Submitted application application_1582510523493_0001
20/02/24 03:09:32 INFO mapreduce.Job: The url to track the job: http://ip-172-31-38-169.ec2.inter
nal:8088/proxy/application_1582510523493_0001/
20/02/24 03:09:32 INFO mapreduce.Job: Running job: job_1582510523493_0001
20/02/24 03:09:38 INFO mapreduce.Job: Job job_1582510523493_0001 running in uber mode : false
20/02/24 03:09:38 INFO mapreduce.Job:  map 0% reduce 0%
20/02/24 03:09:49 INFO mapreduce.Job:  map 16% reduce 0%
20/02/24 03:09:50 INFO mapreduce.Job:  map 30% reduce 0%
20/02/24 03:09:52 INFO mapreduce.Job:  map 41% reduce 0%
20/02/24 03:09:53 INFO mapreduce.Job:  map 47% reduce 0%
20/02/24 03:09:55 INFO mapreduce.Job:  map 53% reduce 0%
20/02/24 03:09:56 INFO mapreduce.Job:  map 60% reduce 0%
20/02/24 03:09:57 INFO mapreduce.Job:  map 77% reduce 0%
20/02/24 03:09:59 INFO mapreduce.Job:  map 81% reduce 0%
20/02/24 03:10:02 INFO mapreduce.Job:  map 83% reduce 0%
20/02/24 03:10:05 INFO mapreduce.Job:  map 100% reduce 0%
20/02/24 03:10:09 INFO mapreduce.Job:  map 100% reduce 100%
20/02/24 03:10:09 INFO mapreduce.Job: Job job_1582510523493_0001 completed successfully
20/02/24 03:10:09 INFO mapreduce.Job: Counters: 49
        File System Counters
```

```
real    0m40.918s
user    0m3.923s
sys     0m0.243s
[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -du /data/wordcount1/
0         /data/wordcount1/_SUCCESS
20056175  /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-38-169 ~]$ 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
```

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

HW1 time:

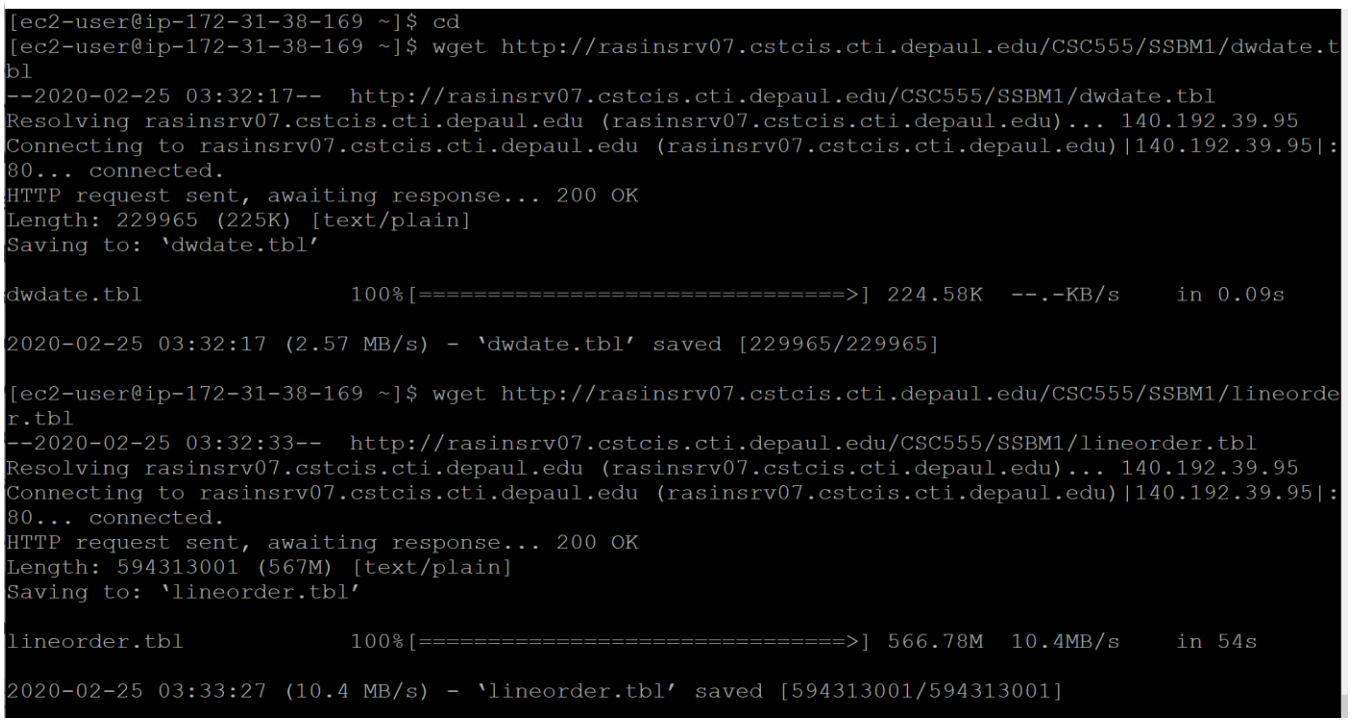
```
real    1m12.273s
user    0m4.018s
sys     0m0.179s
[ec2-user@ip-172-31-41-200 ~]$
```

- So in Homework 1, the job took about 72 seconds to complete and used a single node. In this assignment, the job was split in 2 according to the screenshot above. This would suggest that the job would be completed in about half the time (it should be faster at least). From the screenshot above, it took roughly 41 seconds to complete in this assignment. This is not exactly half the time, but it is faster as expected.

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

*First I have to get the tables: wget (the links from above)



```
[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl
--2020-02-25 03:35:03--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 17139259 (16M) [text/plain]
Saving to: 'part.tbl'

part.tbl                100%[===================================>]  16.34M  9.99MB/s    in 1.6s

2020-02-25 03:35:05 (9.99 MB/s) - 'part.tbl' saved [17139259/17139259]

[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tb
l
--2020-02-25 03:35:25--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tbl
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 166676 (163K) [text/plain]
Saving to: 'supplier.tbl'

supplier.tbl            100%[===================================>] 162.77K  --.-KB/s    in 0.07s

2020-02-25 03:35:25 (2.25 MB/s) - 'supplier.tbl' saved [166676/166676]

[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tb
l
--2020-02-25 03:36:52--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 2837046 (2.7M) [text/plain]
Saving to: 'customer.tbl'

customer.tbl            100%[===================================>]   2.71M  4.22MB/s    in 0.6s

2020-02-25 03:36:55 (4.22 MB/s) - 'customer.tbl' saved [2837046/2837046]

[ec2-user@ip-172-31-38-169 ~]$
```

*Now that the 5 tables are loaded, let's check they're really in:

```
[ec2-user@ip-172-31-38-169 ~]$ ls
bioproject.xml  dwdate.tbl    hadoop-2.6.4.tar  myHadoop.tar  supplier.tbl
customer.tbl    hadoop-2.6.4  lineorder.tbl     part.tbl
```

So they're all in. Following Assignment 2 steps for getting Hive:

```
[ec2-user@ip-172-31-38-169 ~]$ cd
[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hive
-2.0.1-bin.tar.gz
--2020-02-25 05:32:58--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hive-2.0.1-bin.
tar.gz
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.9
5|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 139856338 (133M) [application/x-gzip]
Saving to: ‘apache-hive-2.0.1-bin.tar.gz’

apache-hive-2.0.1-bin.t 100%[============================>] 133.38M  10.4MB/s    in 13s

2020-02-25 05:33:10 (10.4 MB/s) - ‘apache-hive-2.0.1-bin.tar.gz’ saved [139856338/139856338]

[ec2-user@ip-172-31-38-169 ~]$ gunzip apache-hive-2.0.1-bin.tar.gz
[ec2-user@ip-172-31-38-169 ~]$ tar xvf apache-hive-2.0.1-bin.tar
apache-hive-2.0.1-bin/conf/hive-log4j2.properties.template
```

```
[ec2-user@ip-172-31-38-169 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
mkdir: `/tmp': File exists
[ec2-user@ip-172-31-38-169 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
mkdir: `/user/hive/warehouse': File exists
[ec2-user@ip-172-31-38-169 ~]$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
[ec2-user@ip-172-31-38-169 ~]$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -mkdir /user/ec2-user/
mkdir: `/user/ec2-user': File exists
[ec2-user@ip-172-31-38-169 ~]$ cd $HIVE_HOME
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ $HIVE_HOME/bin/schematool -initSchema -dbT
pe derby
which: no hbase in (/home/ec2-user/apache-hive-2.0.1-bin/bin:/usr/local/bin:/bin:/usr/bin:/us
/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/hadoop-2.6.4/bin:/home/ec2-user/hadoo
-2.6.4/sbin:/home/ec2-user/pig-0.15.0/bin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl-2
4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j-l
g4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 2.0.0
Initialization script hive-schema-2.0.0.derby.sql
Initialization script completed
schemaTool completed
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ bin/hive
which: no hbase in (/home/ec2-user/apache-hive-2.0.1-bin/bin:/usr/local/bin:/bin:/usr/bin:/us
/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/hadoop-2.6.4/bin:/home/ec2-user/hadoo
-2.6.4/sbin:/home/ec2-user/pig-0.15.0/bin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
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-l
g4j12-1.7.5.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]

Logging initialized using configuration in jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/
ive-common-2.0.1.jar!/hive-log4j2.properties
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
sing a different execution engine (i.e. tez, spark) or using Hive 1.X releases.
hive>
```

Create tables

```
hive> 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;
OK
Time taken: 1.218 seconds
```

```
hive> 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;
OK
Time taken: 0.078 seconds
```

```
hive> 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;
OK
Time taken: 0.061 seconds
```

```
hive> create table dwdate (
    >   d_datekey            int,
    >   d_date               varchar(19),
    >   d_dayofweek          varchar(10),
    >   d_month              varchar(10),
    >   d_year               int,
    >   d_yearmonthnum       int,
    >   d_yearmonth          varchar(8),
    >   d_daynuminweek       int,
    >   d_daynuminmonth      int,
    >   d_daynuminyear       int,
    >   d_monthnuminyear     int,
    >   d_weeknuminyear      int,
    >   d_sellingseason      varchar(13),
    >   d_lastdayinweekfl    varchar(1),
    >   d_lastdayinmonthfl   varchar(1),
    >   d_holidayfl          varchar(1),
    >   d_weekdayfl          varchar(1)
    > )
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '|' STORED AS TEXTFILE;
OK
Time taken: 0.071 seconds
```

```
hive> 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;
OK
Time taken: 0.068 seconds
```

*Now that the tables have been created, time to load in the data:

```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl' OVERWRITE INTO TABLE part;
Loading data to table default.part
OK
Time taken: 0.773 seconds
hive> select * from part limit 2;
OK
1       lace spring     MFGR#1  MFGR#11 MFGR#1121       goldenrod       PROMO BURNISHED COPP
R       7       JUMBO PKG
2       rosy metallic   MFGR#4  MFGR#43 MFGR#4318       blush   LARGE BRUSHED BRASS     1
G CASE
Time taken: 0.837 seconds, Fetched: 2 row(s)
```

```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl' OVERWRITE INTO TABLE supplier;
Loading data to table default.supplier
OK
Time taken: 0.222 seconds
hive> select * from supplier limit 2;
OK
1       Supplier#000000001      sdrGnXCDRcfriBvY0KL,i   PERU     0      PERU    AMERICA 27-98
9-741-2988
2       Supplier#000000002      TRMhVHz3XiFu    ETHIOPIA 1      ETHIOPIA        AFRICA  15-76
8-687-3665
Time taken: 0.102 seconds, Fetched: 2 row(s)
```

```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;
Loading data to table default.customer
OK
Time taken: 0.231 seconds
hive> select * from customer limit 2;
OK
1       Customer#000000001      j5JsirBM9P      MOROCCO  0      MOROCCO AFRICA  25-989-741-29
88      BUILDING
2       Customer#000000002      487LW1dovn6Q4dMVym      JORDAN   1      JORDAN  MIDDLE EAST 2
3-768-687-3665  AUTOMOBILE
Time taken: 0.093 seconds, Fetched: 2 row(s)
```

```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/dwdate.tbl' OVERWRITE INTO TABLE dwdate;
Loading data to table default.dwdate
OK
Time taken: 0.247 seconds
hive> select * from dwdate limit 2;
OK
19920101        January 1, 1992 Thursday        January 1992    199201  Jan1992 5       1   1
1       1       Winter  0       1       1       1
19920102        January 2, 1992 Friday  January 1992    199201  Jan1992 6       2       2   1
1       Winter  0       1       0       1
Time taken: 0.118 seconds, Fetched: 2 row(s)
```

```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl' OVERWRITE INTO TABLE lineorder;
Loading data to table default.lineorder
OK
Time taken: 7.185 seconds
hive> select * from lineorder limit 2;
OK
1       1       7381    155190  828     19960102        5-LOW   0       17      2116823 17366
547     4       2032150 74711   2       19960212        TRUCK
1       2       7381    67310   163     19960102        5-LOW   0       36      4598316 17366
547     9       4184467 76638   6       19960228        MAIL
Time taken: 0.097 seconds, Fetched: 2 row(s)
```

*So it appears that all 5 tables are now in Hive. Time to try to run the requested Queries

Query 1.2 * Time taken: 33.784 seconds**

```
--Q1.2  Simplified to remove expression in sum
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;
```

```
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. tez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20200225060008_d6da0c36-cf30-4437-9f80-a9dd9336294b
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
.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
og4j12-1.7.5.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_20200225060008_d6da0c36-cf30-4437-9f80-a9dd933629
log
2020-02-25 06:00:14     Starting to launch local task to process map join;      maximum mem
```

```
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2020-02-25 06:00:24,349 Stage-2 map = 0%,  reduce = 0%
2020-02-25 06:00:35,139 Stage-2 map = 78%,  reduce = 0%, Cumulative CPU 12.76 sec
2020-02-25 06:00:36,172 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 13.06 sec
2020-02-25 06:00:41,436 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 14.48 sec
MapReduce Total cumulative CPU time: 14 seconds 480 msec
Ended Job = job 1582510523493 0002
MapReduce Jobs Launched:
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 14.48 sec   HDFS Read: 594368462 HDFS Writ
e: 12 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 480 msec
OK
14215822897
Time taken: 33.784 seconds, Fetched: 1 row(s)
```

Query 1.3 * Time taken: 29.558 seconds**

```
--Q1.3  Simplified to remove expression in sum
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;
```

```
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. tez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20200225060204_fda8b8e0-acdd-4a77-8ca6-172a29e4e3e7
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-l
og4j12-1.7.5.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_20200225060204_fda8b8e0-acdd-4a77-8ca6-172a29e4e3e7.
log
2020-02-25 06:02:10     Starting to launch local task to process map join;      maximum memor
```

```
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2020-02-25 06:02:17,749 Stage-2 map = 0%,  reduce = 0%
2020-02-25 06:02:27,100 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 5.28 sec
2020-02-25 06:02:28,139 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 13.75 sec
2020-02-25 06:02:33,292 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 15.18 sec
MapReduce Total cumulative CPU time: 15 seconds 180 msec
Ended Job = job_1582510523493_0003
MapReduce Jobs Launched:
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 15.18 sec   HDFS Read: 594368581 HDFS Writ
e: 11 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 180 msec
OK
4435791464
Time taken: 29.558 seconds, Fetched: 1 row(s)
hive>
```

Query 2.1 * Time taken: 113.418 seconds**

```
--Q2.1  No simpifications
select sum(lo_revenue), d_year, p_brand1
from lineorder, dwdate, part, supplier
where lo_orderdate = d_datekey
  and lo_partkey = p_partkey
  and lo_suppkey = s_suppkey
  and p_category = 'MFGR#12'
  and s_region = 'AMERICA'
group by d_year, p_brand1
order by d_year, p_brand1;
```

```
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. tez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20200225060528_e06df259-6ef8-448e-ba4e-d62a32797073
Total jobs = 6
```

```
Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1
2020-02-25 06:07:08,533 Stage-5 map = 0%,  reduce = 0%
2020-02-25 06:07:14,720 Stage-5 map = 100%,  reduce = 0%, Cumulative CPU 0.93 sec
2020-02-25 06:07:20,921 Stage-5 map = 100%,  reduce = 100%, Cumulative CPU 2.23 sec
MapReduce Total cumulative CPU time: 2 seconds 230 msec
Ended Job = job_1582510523493_0007
MapReduce Jobs Launched:
Stage-Stage-13: Map: 3   Cumulative CPU: 30.51 sec   HDFS Read: 594359190 HDFS Write: 1847332
91 SUCCESS
Stage-Stage-10: Map: 2   Cumulative CPU: 16.64 sec   HDFS Read: 184745042 HDFS Write: 8750831
 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 4.67 sec   HDFS Read: 8763341 HDFS Write:
9913 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 2.23 sec   HDFS Read: 15713 HDFS Write: 69
37 SUCCESS
Total MapReduce CPU Time Spent: 54 seconds 50 msec
OK
567838207       1992    MFGR#121
610663790       1992    MFGR#1210
550769662       1992    MFGR#1211
649205856       1992    MFGR#1212
```

```
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: 113.418 seconds, Fetched: 280 row(s)
```

Next STEP:

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 6 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 CREATE TABLE statement of the target table accordingly (since you are introducing a longer column).

*started off by making the customer.py transformer

```
[ec2-user@ip-172-31-38-169 ~]$ nano customer.py
[ec2-user@ip-172-31-38-169 ~]$ ls
apache-hive-2.0.1-bin       customer.py   hadoop-2.6.4       myHadoop.tar
apache-hive-2.0.1-bin.tar   customer.tbl   hadoop-2.6.4.tar   part.tbl
bioproject.xml              dwdate.tbl    lineorder.tbl      supplier.tbl
```

```
  GNU nano 2.5.3                File: customer.py                                Modified

#!/usr/bin/python
import sys

for line in sys.stdin:
                line = line.strip().split(',')
                line[2] = line[2][:6]
                line[3] = line[3][:len(line[3])-1] + ' ' + '#' + line[3][len(line[3])-1]
                print '\t'.join(line)
```

*Then getting back into Hive:

```
[ec2-user@ip-172-31-38-169 ~]$ cd $HIVE_HOME
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ $HIVE_HOME/bin/schematool -initSchema -d
ype derby
which: no hbase in (/home/ec2-user/apache-hive-2.0.1-bin/bin:/usr/local/bin:/bin:/usr/bin:/
r/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/hadoop-2.6.4/bin:/home/ec2-user/ha
op-2.6.4/sbin:/home/ec2-user/pig-0.15.0/bin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl
.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j
og4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
org.apache.hadoop.hive.metastore.HiveMetaException: Failed to get schema version.
*** schemaTool failed ***
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ rm -rf metastore_db/
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ $HIVE_HOME/bin/schematool -initSchema -d
ype derby
which: no hbase in (/home/ec2-user/apache-hive-2.0.1-bin/bin:/usr/local/bin:/bin:/usr/bin:/
r/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/hadoop-2.6.4/bin:/home/ec2-user/ha
op-2.6.4/sbin:/home/ec2-user/pig-0.15.0/bin:/home/ec2-user/.local/bin:/home/ec2-user/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ec2-user/apache-hive-2.0.1-bin/lib/log4j-slf4j-impl
.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ec2-user/hadoop-2.6.4/share/hadoop/common/lib/slf4j
og4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 2.0.0
Initialization script hive-schema-2.0.0.derby.sql
Initialization script completed
schemaTool completed
[ec2-user@ip-172-31-38-169 apache-hive-2.0.1-bin]$ bin/hive
```

*Customer table is no longer there so re-run create customer:

```
hive> 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;
OK
Time taken: 1.173 seconds
hive> create table customer2 (
    >   c_custkey     INT,
    >   c_address     VARCHAR (6),
    >   c_city        VARCHAR (12)
    > )
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '\t' STORED AS TEXTFILE;
OK
Time taken: 0.067 seconds
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;
Loading data to table default.customer
OK
Time taken: 0.673 seconds
hive> select * from customer limit 1;
OK
1       Customer#000000001      j5JsirBM9P      MOROCCO  0      MOROCCO AFRICA  25-989-741-2988 BUIL
DING
Time taken: 0.819 seconds, Fetched: 1 row(s)
hive>
```

```
hive> add file /home/ec2-user/customer.py;
Added resources: [/home/ec2-user/customer.py]
```

```
Added resources: [/home/ec2-user/customer.py]
hive> 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_address, c_city) FROM customer;
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. tez, spark) or using Hive 1.X releases.
Query ID = ec2-user_20200225074739_c23759c2-be71-40c0-8d2c-32dbd88cd2c2
Total jobs = 3
Launching Job 1 out of 3
```

select c_custkey, c_address, c_city from customer limit 20;

OK

1 j5JsirBM9P MOROCCO #0

2 487LW1dovn6Q4dMVym JORDAN #1

3 fkRGN8n ARGENTINA #7

4 4u58h f EGYPT #4

5 hwBtxkoBF qSW4KrI CANADA #5

6 g1s,pzDenUEBW3O,2 pxu SAUDI ARA #2

7 8OkMVLQ1dK6Mbu6WG9 CHINA #0

8 j,pZ,Qp,qtFEo0r0c 92qo PERU #6

9 vglql8H6zoyuLMFN INDIA #6

10 Vf mQ6Ug9Ucf5OKGYq fs ETHIOPIA #9

11 cG48rYjF3Aw7xs UNITED KI #3

12 Sb4gxKs7 JORDAN #5

13 Ez3ax0D5HnUbeU CANADA #8

14 h3GFMzeFf ARGENTINA0

15 3y4KK4CcfNwNCTP0u0p1Rk6 UNITED KI #0

16 P2IQMff18er IRAN #5

17 Js JrVHNAyCYMANzPGzvonS BRAZIL #6

18 YyukcsqIxlyuXs7 FRANCE #0

19 yO0XPkiuSWk0vN Ffc CHINA #3

20 i bGScA RUSSIA #0

Time taken: 0.084 seconds, Fetched: 20 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).

```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -put lineorder.tbl /user/ec2-user/
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -ls /user/ec2-user
Found 1 items
-rw-r--r--   2 ec2-user supergroup  594313001 2020-02-25 08:10 /user/ec2-user/lineord
er.tbl
```

Query 0.1 ***Time taken = 1 minute 25 seconds

0.1 Added simple test query

```
SELECT AVG(lo_revenue)
FROM lineorder;
```

```
  GNU nano 2.5.3                File: pig_script.pig                      Modified

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;
```

```
[ec2-user@ip-172-31-38-169 ~]$ cd $PIG_HOME
[ec2-user@ip-172-31-38-169 pig-0.15.0]$ bin/pig -f pig_script.pig
20/02/25 08:12:17 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
20/02/25 08:12:17 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
20/02/25 08:12:17 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
```

```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.4   0.15.0  ec2-user        2020-02-25 08:12:19     2020-02-25 08:13:41     GROUP
_BY

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime
MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   Featu
re      Outputs
job_1582510523493_0020  5       1       69      33      54      53      34      34  3
4       34      AVG,Grouped,lineorder   GROUP_BY,COMBINER       hdfs://172.31.38.169/
tmp/temp5075393/tmp794916561,

Input(s):
Successfully read 6001215 records (594331260 bytes) from: "/user/ec2-user/lineorder.t
bl"

Output(s):
Successfully stored 1 records (13 bytes) in: "hdfs://172.31.38.169/tmp/temp5075393/tm
p794916561"

Counters:
Total records written : 1
Total bytes written : 13
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 10
Total records proactively spilled: 9363686

Job DAG:
job_1582510523493_0020
```

```
2020-02-25 08:13:42,297 [main] INFO  org.apache.pig.Main - Pig script completed in 1
minute, 25 seconds and 182 milliseconds (85182 ms)
```

Query 0.1 Result: 3634300.7095

Query 0.2 Time Taken = 4 seconds*

```
--Q0.2 Added simple test query
SELECT lo_discount, COUNT(lo_extendedprice)
FROM lineorder
GROUP BY lo_discount;
```

```
Grouped = Group lineorder BY lo_discount;
Counted = FOREACH Grouped GENERATE Group AS lo_discount, COUNT(lineorder.lo_extendedprice) as cnt;
DUMP Counted;
```

```
2020-02-25 08:31:07,966 [main] INFO  org.apache.pig.Main - Pig script completed in 4 seconds and 425 milliseconds (4425 ms)
[ec2-user@ip-172-31-38-169 pig-0.15.0]$
```

***This runtime is a bit off. I ran the code in bin/pig first to get the results and then went back to check runtimes. It took longer than 4 seconds on the first go.

Query 0.2 Result: (see below)

```
ocess : 1
(0,544886)
(1,545834)
(2,546173)
(3,545293)
(4,545545)
(5,546395)
(6,544970)
(7,546192)
(8,544803)
(9,545309)
(10,545815)
grunt>
```

Query 0.3 Time Taken = 2 seconds*

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

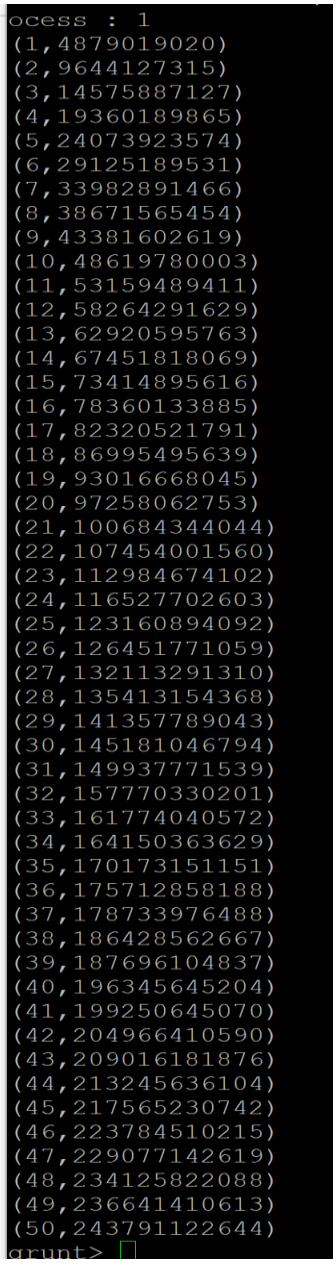
```
);

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;
```

```
Details at logfile: /home/ec2-user/pig-0.15.0/pig_1582619697990.log
2020-02-25 08:34:59,979 [main] INFO  org.apache.pig.Main - Pig script completed in 2 seconds and 67 milliseconds (2067 ms)
```

***Again, it was longer than 2 seconds for the runtime. I already ran it and it took longer.

Query 0.3 Result: (see below)



*runtimes are pretty fast overall for these queries. In one of them I was also loading in the data table for lineorder so that first one took longer.

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 BETWEEN 3 AND 5
GROUP BY lo_quantity;
```

*Setup

```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -mkdir /user/ec2-user/streaming
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -put lineorder.tbl streaming
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -ls streaming
Found 1 items
-rw-r--r--   2 ec2-user supergroup  594313001 2020-02-25 08:54 streaming/lineorder.tbl
[ec2-user@ip-172-31-38-169 ~]$
```

```
[ec2-user@ip-172-31-38-169 pig-0.15.0]$ cd
[ec2-user@ip-172-31-38-169 ~]$ nano mapper03.py
[ec2-user@ip-172-31-38-169 ~]$ nano reducer03.py
[ec2-user@ip-172-31-38-169 ~]$ ls
apache-hive-2.0.1-bin      customer.py           dwdate.tbl        lineorder.tbl  part.tbl        reducer03.py
apache-hive-2.0.1-bin.tar  customer.tbl          hadoop-2.6.4      mapper03.py    pig-0.15.0      supplier.tbl
bioproject.xml             customer_transform.py hadoop-2.6.4.tar  myHadoop.tar   pig-0.15.0.tar
[ec2-user@ip-172-31-38-169 ~]$
```

ec2-user@ip-172-31-38-169:~

```
 GNU nano 2.5.3                                File: mapper03.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)
```

```
  GNU nano 2.5.3                                File: reducer03.py

#!/usr/bin/python
import sys

current_id = None
current_total = 0
id = None

for line in sys.stdin:
        line = line.strip()
        ln = line.split('\t')
        id = ln[0]
        val = int(ln[1])
        if current_id == id:
                current_total += val
        else:
                if current_id:
                        print '%s\t%d' % (current_id, current_total)
                current_id = id
                current_total = val
if current_id == id:
    print '%s\t%d' % (current_id, current_total)
```

hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input /user/ec2-user/streaming -output /data/streaming03 -mapper mapper03.py -reducer reducer03.py -file mapper03.py -file reducer03.py

```
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ nano mapper03.py
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ nano reducer03.py
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ ls
bin  etc  include  lib  libexec  LICENSE.txt  logs  mapper03.py  NOTICE.txt  README.txt  reducer03.py  sbin  share
```

```
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input /user/ec2-user
/streaming -output /data/streaming03 -mapper mapper03.py -reducer reducer03.py -file mapper03.py -file reducer03.py
20/02/25 09:35:16 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper03.py, reducer03.py, /tmp/hadoop-unjar438293594151897780/] [] /tmp/streamjob463723862520544522.jar tmp
Dir=null
20/02/25 09:35:17 INFO client.RMProxy: Connecting to ResourceManager at /172.31.38.169:8032
20/02/25 09:35:18 INFO client.RMProxy: Connecting to ResourceManager at /172.31.38.169:8032
20/02/25 09:35:18 INFO mapred.FileInputFormat: Total input paths to process : 1
20/02/25 09:35:18 INFO mapreduce.JobSubmitter: number of splits:5
20/02/25 09:35:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1582510523493_0024
20/02/25 09:35:19 INFO impl.YarnClientImpl: Submitted application application_1582510523493_0024
20/02/25 09:35:19 INFO mapreduce.Job: The url to track the job: http://ip-172-31-38-169.ec2.internal:8088/proxy/application_
1582510523493_0024/
20/02/25 09:35:19 INFO mapreduce.Job: Running job: job_1582510523493_0024
20/02/25 09:35:24 INFO mapreduce.Job: Job job_1582510523493_0024 running in uber mode : false
20/02/25 09:35:24 INFO mapreduce.Job:  map 0% reduce 0%
20/02/25 09:35:34 INFO mapreduce.Job:  map 20% reduce 0%
20/02/25 09:35:36 INFO mapreduce.Job:  map 40% reduce 0%
20/02/25 09:35:38 INFO mapreduce.Job:  map 57% reduce 0%
20/02/25 09:35:39 INFO mapreduce.Job:  map 68% reduce 0%
20/02/25 09:35:40 INFO mapreduce.Job:  map 77% reduce 0%
```

```
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=6
                Launched reduce tasks=1
                Data-local map tasks=6
                Total time spent by all maps in occupied slots (ms)=62797
                Total time spent by all reduces in occupied slots (ms)=10083
                Total time spent by all map tasks (ms)=62797
                Total time spent by all reduce tasks (ms)=10083
                Total vcore-milliseconds taken by all map tasks=62797
                Total vcore-milliseconds taken by all reduce tasks=10083
                Total megabyte-milliseconds taken by all map tasks=64304128
                Total megabyte-milliseconds taken by all reduce tasks=10324992
        Map-Reduce Framework
                Map input records=6001215
                Map output records=1636893
                Map output bytes=17499115
                Map output materialized bytes=20772931
                Input split bytes=550
                Combine input records=0
                Combine output records=0
                Reduce input groups=50
                Reduce shuffle bytes=20772931
                Reduce input records=1636893
                Reduce output records=50
                Spilled Records=3273786
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=453
                CPU time spent (ms)=23550
                Physical memory (bytes) snapshot=1537519616
                Virtual memory (bytes) snapshot=5941633024
                Total committed heap usage (bytes)=1084227584
        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=594329385
        File Output Format Counters
                Bytes Written=769
0/02/25 09:35:47 INFO streaming.StreamJob: Output directory: /data/streaming03
ec2-user@ip-172-31-38-169 hadoop-2.6.4]$
```

```
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ hadoop fs -cat /data/streaming03/part-00000
1       4879019020
10      48619780003
11      53159489411
12      58264291629
13      62920595763
14      67451818069
```

Query 0.3 Results (below):

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.