CS553 Programming Assignment #2
Sort on Hadoop/Spark

Problem:

This programming assignment covers the Sort application implemented in 3 different ways: Java, Hadoop, and Spark. The sorting application can read large file, which is larger than system memory size and sort it in place. This assignment uses 2 datasets, 10 GB and 100GB dataset to benchmark the 3 approaches to sorting.

Methodology:

Shared-Memory Sort: This program sorts data files of any size, which are larger than the memory. This program is implemented using Java language. This program implements external sorting algorithm that can handle massive amount of data. External sorting is used when data file is larger than main memory of computing device instead it uses external memory .i.e. hard drive. This program uses multi-threading to take advantage of multi-core.

Shared-Memory external sorting program steps

1. Read input data file path and calculate size of file.
2. Find available main memory size of computing device.
3. Calculate data block size that fits into main memory
4. Read block size amount of data from input file and save it in external memory.
5. Follow step 4 up to end of input file.
6. Initialize the thread-pool with number of threads.
7. Assign each thread in thread pool with task in step 8 using executors.
8. Read each split file from step 4 and sort it using any n log n algorithm, this program using merge sort algorithm to sort array of strings.
9. Save sorted file in external memory.
10. Merge sorted files in external memory and write it to output file.

CS553 Programming Assignment #2
Sort on Hadoop/Spark

Hadoop Sort: This program is implemented in java based on MapReduce programming model. This consists of map and reduce function which extends Hadoop mapper and reducer class respectively.

Implementation details of Hadoop sort are as follows.

1. Implement map function, which takes Key and value input as text format.
2. Implement reduce function, which takes key as text and List of values as value.
3. Set InputFormatClass as KeyValueTextInputFormat, This class brakes file into line and split each line into key and value based on separator.
4. Set mapper class to map function implemented in step1
5. Set reducer class to reduce function implemented in step2
6. Set OutputKeyClass and OutputValueClass as text.
7. Set FileInputFormatClass to input file path.
8. Set OutputFormatClass to output file path.
9. Run the sorting job and wait for completion.
10. Map function in step1 take its file each record as input in the form of key and value based on separator. Map functions outputs data file each record in form of key and value.
11. Hadoop framework implements sorting and shuffling after map function.
12. Reduce function take input from map function and gives output as key and value in sorted order.
13. Write sorted records to output file path.

Write a description of the function of each file, and what modifications you had to make to go from 1 node to multiple nodes

1) conf/master: This file lists IP address of machine runs secondary name nodes. Master file in master contains localhost.

2) conf/slaves: slave file have localhost entry by default. For master instance add IP address of master node and all slave node. For slave node add IP of its own in slaves file. This file is used to run data nodes and task trackers on slaves.

CS553 Programming Assignment #2
Sort on Hadoop/Spark

3) conf/core-site.xml: This file sets HDFS file system path and Hadoop temporary storage. This file informs Hadoop about name node execution. It contains configuration settings for HDFS I/O and port number to which name node daemon runs and listen.

Structure of file as follows

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>
 <property>
  <name>fs.default.name</name>
  <value>hdfs://MasterIP(ec2-52-87-241-215.compute-1.amazonaws.com:54310)
</value>
 </property>
</configuration>
```

4) conf/hdfs-site.xml: This file is used to configure HDFS daemons such as Name node, Data node and secondary name node. This file is used to set data block replication factor .i.e. number data replication per data block and setting HDFS permissions using Boolean value like true to enable permission, false to disable.

Structure of this file is as follows

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>
 <property>
   <name>dfs.permissions</name>
   <value>false</value>
 </property>
```

CS553 Programming Assignment #2
Sort on Hadoop/Spark

```
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

5) conf/mapred-site.xml: This file is used to configure map reduce framework and job tracker, task tracker. Map reduce job tracker runs at IP and port mentioned in mapreduce.hobtracker.address property. File can set map reduce framework name. This file is used to set different properties like mapred.system.dir, mapred.local.dir, mapred.tasktracker.{map|reduce}.tasks.maximum, fs.hosts/dfs.hosts.ex, mapred.hosts/mapred,

1) What is a Master node? What is a Slaves node?

Master node manages all other node such as data node, secondary nodes in cluster. Master node host various storage and processing services. Master manages following nodes to achieve fault tolerant

Name node: it manages HDFS storage.

Secondary name node:

Resource manager: This manages scheduling of tasks of Hadoop. It runs by yarn frame work.

Job tracker: It is used in 1 server job tracking.

Slave node is used to store and process data in Hadoop cluster. The different components involved in storage and computing of data are as follows.

CS553 Programming Assignment #2
Sort on Hadoop/Spark

Node manager: It manages individual slave node and reports back to resource manager.

Data node: This node is used to store the data in Hadoop cluster.

2) Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?

Unique ports are used to assign each port with unique functionality otherwise if we use same port for all services then all functionality get interrupt and communication problem between master and slave node. For example we use different port for HDFS filesystem, job tracker and Resource tracker.

3) How can we change the number of mappers and reducers from the configuration file?

Number of mappers and reducers can changed by setting "mapred.tasktracker.map.tasks.maximum" and "mapred.tasktracker.reduce.tasks.maximum" properties in mapred-site.xml

Spark Sort:

This program is implemented using java. Sparks provides API to read data into sparks data structure called RDD (resilient distributed dataset). The map and reduce function are implemented on RDD. The in-memory primitives of sparks provides better performance compared to any other map-reduce framework.

Implementation details of Spark sort are as follows.

1. Read input data file into sparks RDD structure.
2. Implement Pair function which takes string as input and outputs key and value string as output. Pair function splits input string into two strings by splitting first 10 characters as key and remaining 90 characters as values. Pair function returns set of pair records.

CS553 Programming Assignment #2
Sort on Hadoop/Spark

3. Call sort by key on set of Pair records returned by step2 Pair function. This call sorts the set of pair records by key in ascending order.
4. The sorted set of record pair consists of parenthesis in each record. Remove parenthesis in each record by calling map function.
5. Save the sorted record set in text file format.
6. Parnellism and collect API calls can be used to collect parts of data into single dataset and save as single output text file.

Runtime environment settings

1. **Shared Memory.**
    1. Login to Amazon AWS account.
    2. Launch c3.large instance using Ubuntu operating system AMI.
    3. Install JDK on instance
    4. Copy shared memory program to c3.large instance.
    5. Compile program using javac compiler
    6. Run program using java command

2. **Hadoop.**
    1. **1-Node**
        1. Login to Amazon AWS account
        2. Launch c3.large instance using Ubuntu operating system AMI.
        3. Install updates
        4. Install JDK on instance
        5. Install SSH
        6. Create and set-up SSH certificates to access its nodes, in case of 1 node localhost is its node.
        7. Add newly created SSH certificate key to authorized keys.
        8. Download Hadoop
        9. Extract and move Hadoop folder /usr/local/Hadoop
        10. Create name node and data node directory
        11. Set up following configuration files.
            a. ~./bashrc

CS553 Programming Assignment #2
Sort on Hadoop/Spark

Add following variables

export JAVA_HOME=JDK path (ex. /usr/lib/jvm/java-7-openjdk-amd64)
export HADOOP_INSTALL=Hadoop path (/usr/local/Hadoop)
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"

b. Hadoop-env.sh

Set below variable

export JAVA_HOME=JDK Path (/usr/lib/jvm/java-7-openjdk-amd64)

c. core-site.xml

Create Hadoop temporary directory like /app/Hadoop/temp. Add below configuration in core-site.xml

```
<configuration>
<property>
 <name>hadoop.tmp.dir</name>
 <value>Directory Path (like /app/Hadoop/temp)</value>
</property>
 <property>
 <name>fs.default.name</name>
 <value>Instance Public DNS of instance</value>
</property>
</configuration>
```

CS553 Programming Assignment #2
Sort on Hadoop/Spark

d. mapred-site.xml

Add below configuration

```
<configuration>
<property>
 <name>mapreduce.jobtracker.address</name>
 <value>hdfs://Public DNS of master instance</value>
</property>
<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
</property>
</configuration>
```

e. hdfs-site.xml

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
 <property>
   <name>dfs.permissions</name>
   <value>false</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/Name  node directory (ex
usr/local/hadoop_store/hdfs/namenode) </value>
 </property>
```

CS553 Programming Assignment #2
Sort on Hadoop/Spark

```
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/Datanode directory (ex.
usr/local/hadoop_store/hdfs/datanode) </value>
 </property>
</configuration>
```

12. Format HDFS file system
13. Start Hadoop by running start-all.sh
14. Run jps command to check Hadoop status like node manager, name node, resource manger , data node.

Command for above steps are mentioned in attached Hadoop_install script.

2. **Hadoop 17 node**.
    1. Follow steps mentioned in 1-node Hadoop set-up
    2. Add PEM key using ssh-add command to secured communication between master and nodes.
    3. Take the image of above instance
    4. Launch other 16 c3.large instance using above image AMI.
    5. Open slaves file in ../hadoop/etc/hadoop/slaves in master. Add master public DNS IP and 16 slave node public DNS IP.
    6. Open slaves file in ../hadoop/etc/hadoop/slaves in each slave and add its own public DNS IP.
    7. Start Hadoop in master by running ./start-all.sh

CS553 Programming Assignment #2
Sort on Hadoop/Spark

3. **Spark**
    1. **1 -Node.**
        1. Install 1- node Hadoop as mentioned above.
        2. Download spark and extract
        3. Can change sparks runtime setting in spark-defaults.conf file

           Ex. spark.driver.memory 2g
           spark.executor.extraJavaOptions -XX:+UseConcMarkSweepGC
           spark.driver.maxResultSize 1500g
           spark.akka.frameSize 1000
           spark.default.parallelism 100
           spark.akka.timeout 100

        4. Run start-all.sh in sparks sbin directory.
        5. Run jps to check running status of sparks like worker node, master node.

    2. **17- Node**
        1. Create AMI of required storage space using mount command to club all space in instance.
        2. Move to ec2 directory in spark directory
        3. Set AWS_ACCESS_KEY_ID using export command.
        4. Set AWS_SECRET_ACCESS_KEY using export command.
        5. Run ./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> launch <cluster-name> to launch 17 node cluster.
           <keypair> is name of EC2 key pair
           <key-file> is the private key file for your key pair
           <num-slaves> is the number of slave nodes to launch
           <cluster-name> is the name of cluster.
        6. Run ./spark-ec2 -k <keypair> -i <key-file> login <cluster-name> to login to cluster.
           where <keypair> is name of EC2 pair.
           <key-file> is private key file.

CS553 Programming Assignment #2
Sort on Hadoop/Spark

7. Run ./spark-submit <key-file> <jar-file> <hdfs input file path> <hdfs output file path>

Difficulties

1. Spark 17 node cluster set up. Using of command Run ./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> launch <cluster-name> to launch 17 node set up fails with error connection refused or Public key access denied.This error occurs randomly.

OS used (Linux distribution, kernel): Ubuntu 14.04.3 LTS, Release: 14.04, Codename: trusty

ANT version: Apache Ant(TM) version 1.9.3 compiled on April 8 2014

Java version: java version "1.7.0_95"

Hadoop version: 2.7.2

Spark version: 1.6.1

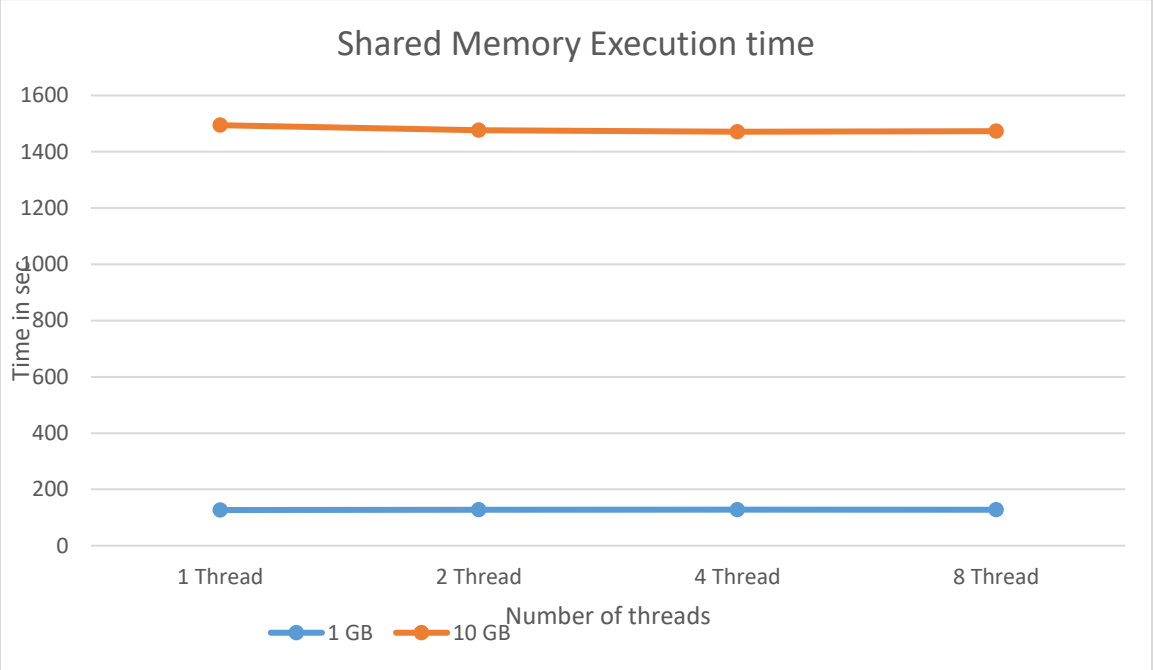
Draw graphs showing execution time data and speedup data. Please explain your results. Please include a comparison between the Shared Memory Sort compared to the Hadoop Sort and Spark Sort for 1 node and Hadoop and Spark at 16 nodes. Can you explain the difference in performance?

Performance

Shared Memory: Execution Time (sec)

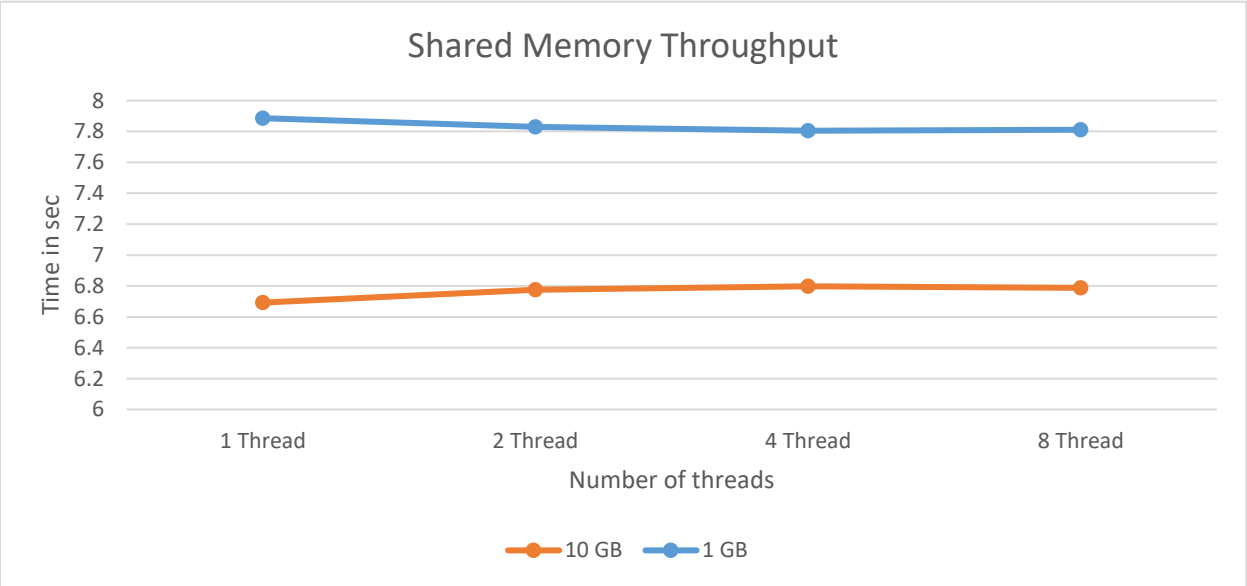
	1 Thread	2 Thread	4 Thread	8 Thread
1 GB	126.802	127.716	128.118	128.006
10 GB	1494.288	1476.089	1471.159	1473.229

CS553 Programming Assignment #2
Sort on Hadoop/Spark



Throughput (mb/sec)

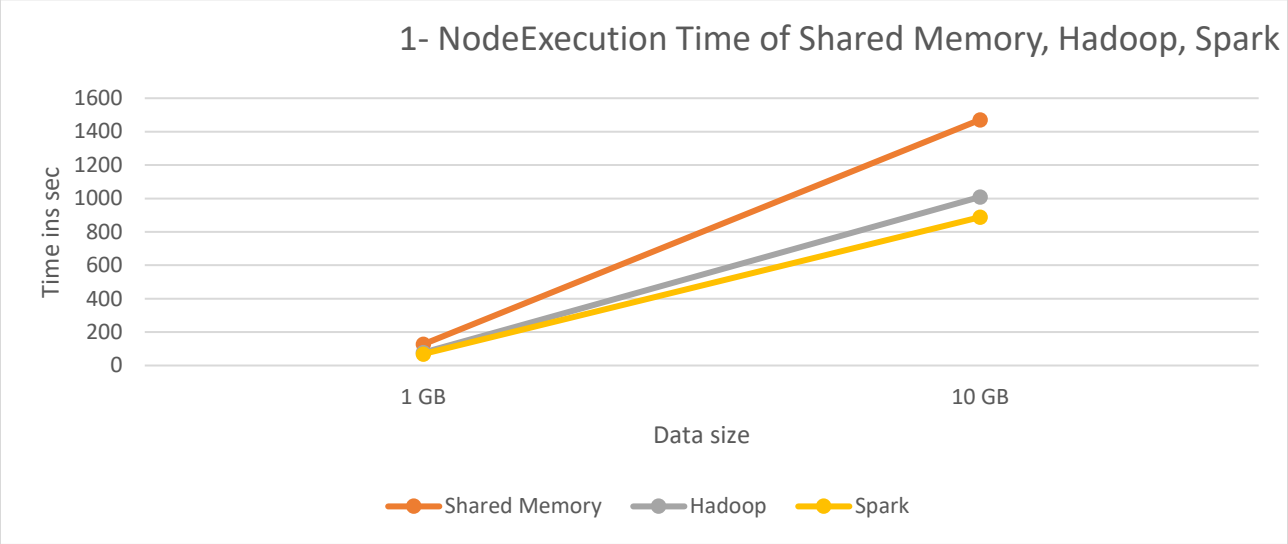
	1 Thread	2 Thread	4 Thread	8 Thread
1 GB	7.886	7.829	7.805	7.8121
10 GB	6.692	6.775	6.7974	6.7878



CS553 Programming Assignment #2
Sort on Hadoop/Spark

1-Node execution time Shared memory, Hadoop, Spark.

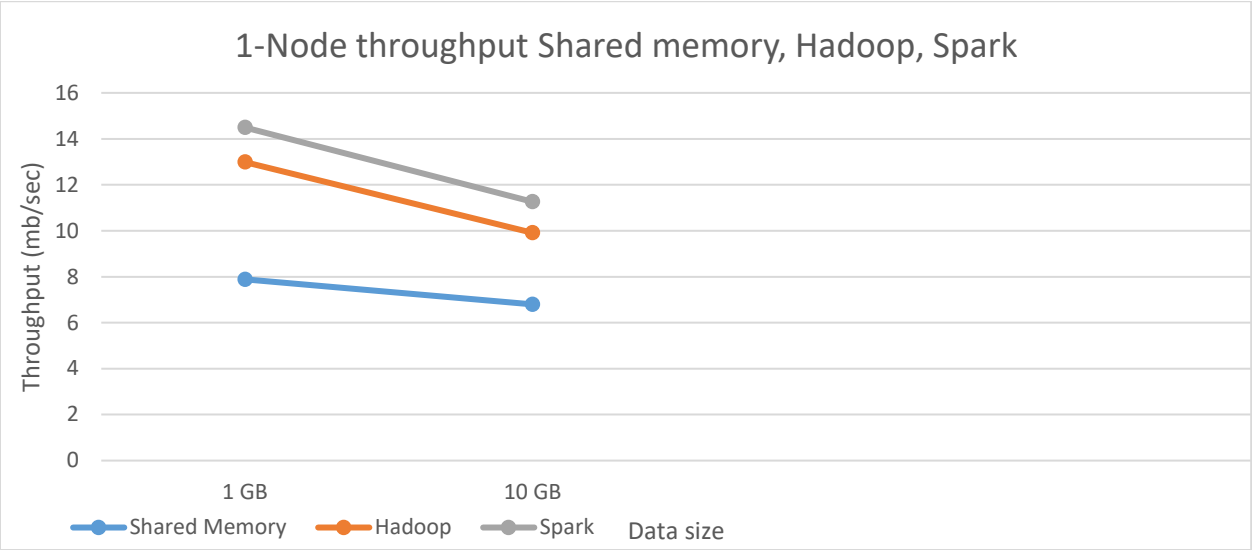
	Shared Memory	Hadoop	Spark
1 GB	126.802	77	69
10 GB	1471.159	1009	888



1-Node Throughput Shared memory, Hadoop, Spark

	Shared Memory	Hadoop	Spark
1 GB	7.886	12.987	14.493
10 GB	6.7974	9.911	11.2613

CS553 Programming Assignment #2
Sort on Hadoop/Spark



Comparison between shared memory, Hadoop, spark on 1 node

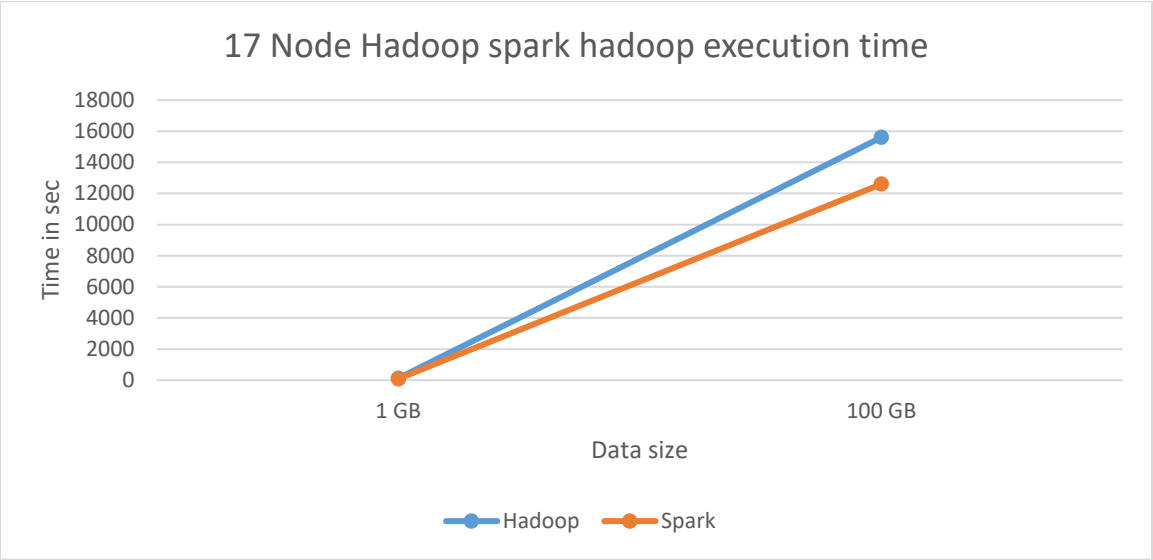
In all experiments, Spark is more efficient than Hadoop, shared memory because Spark in-memory primitive make it faster than Hadoop. Since my shared memory is not optimized code, Spark and Hadoop are faster i.e. taking less time to sort data than shared memory time.

If shared memory code is optimized then it will be faster than Hadoop, spark because both Hadoop, spark takes some extra time schedule tasks and sorts data.

17- Node Hadoop, Spark execution time

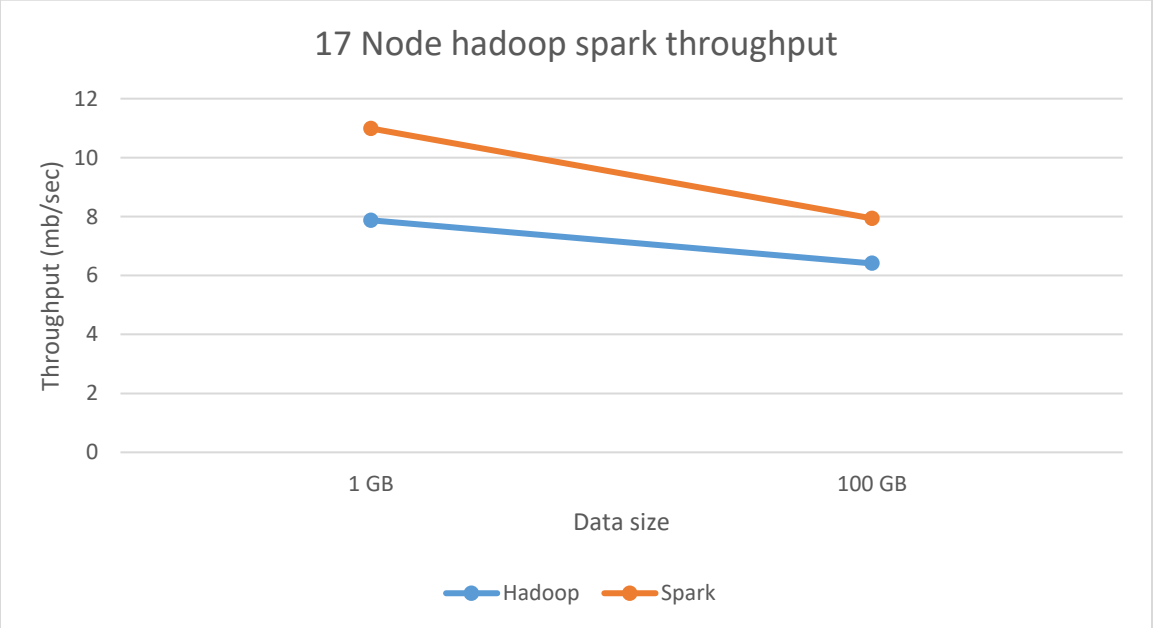
	Hadoop	Spark
1 GB	127	91
100 GB	15600	12600

CS553 Programming Assignment #2
Sort on Hadoop/Spark



17 node Hadoop, spark throughput.

	Hadoop	Spark
1 GB	7.874	10.989
100 GB	6.41	7.9365



CS553 Programming Assignment #2
Sort on Hadoop/Spark

Comparison between 1GB Shared memory at 1 node to Hadoop, Spark at 16 node

	Time (ms)
Shared Memory	126
Hadoop	127
Spark	91

Shared memory performance at 1 node is better than hadoop and spark performance at 16 node because in 16 node spark and Hadoop framework take time to schedule tasks, transfer data block between master and slave node, collect sorted data from master.

Comparison between Hadoop, spark on 16 Node.

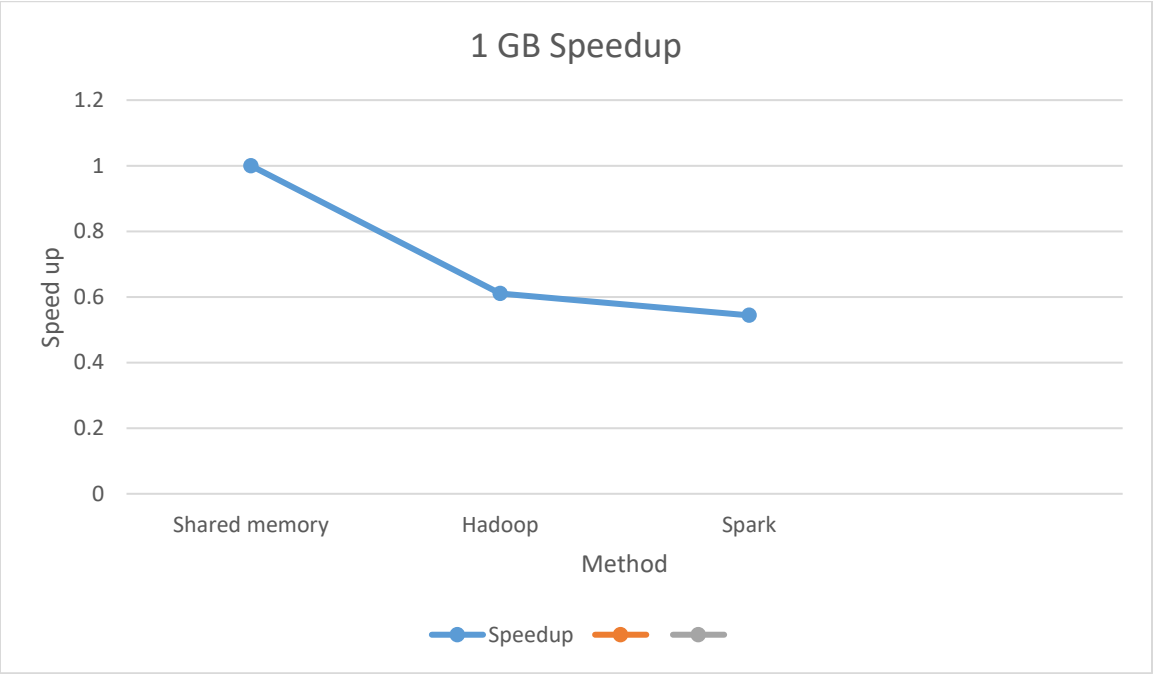
Spark performance is better than Hadoop in 16 node cluster because sparks API structure RDD (Resilient distributed data) is in-memory primitive so it sorts record more efficient than Hadoop. Hadoop splits data into block and stores each block of data as sorted file and then merges each sorted file so it takes more time than spark.

Speedup

1. 1 GB 1 Node Shared memory, Hadoop, Spark.

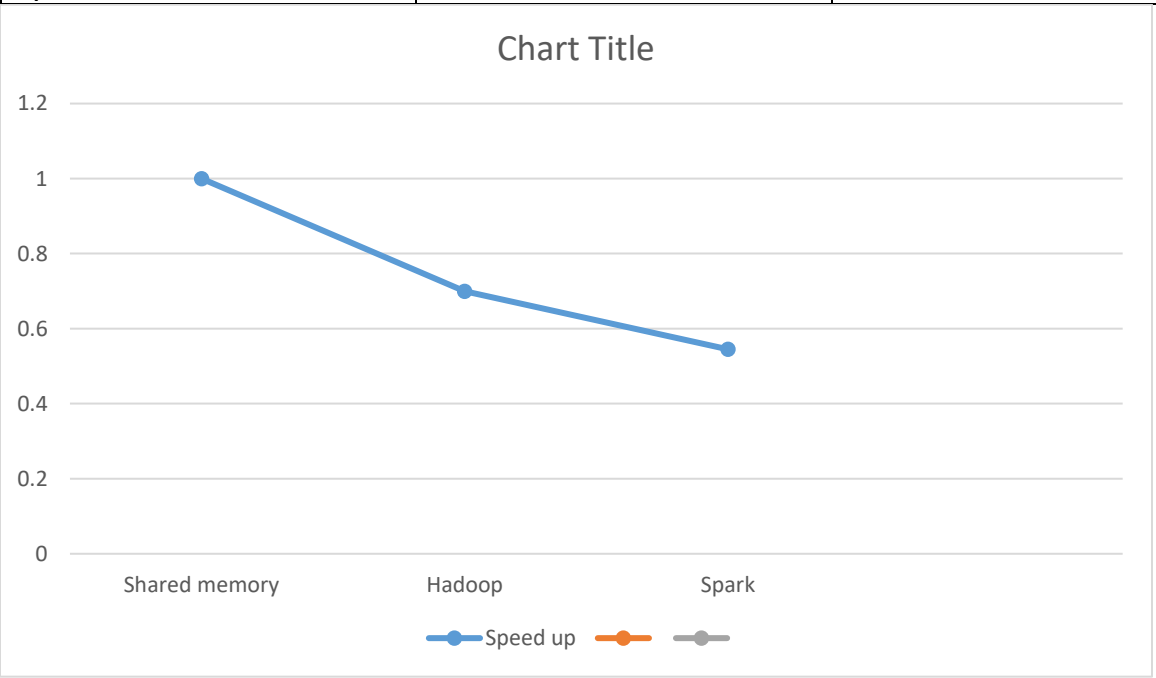
	Speedup	Time (in sec)
Shared memory	1	126.802
Hadoop	0.611	77
Spark	0.544	69

CS553 Programming Assignment #2
Sort on Hadoop/Spark



1 Node 10 GB speed up

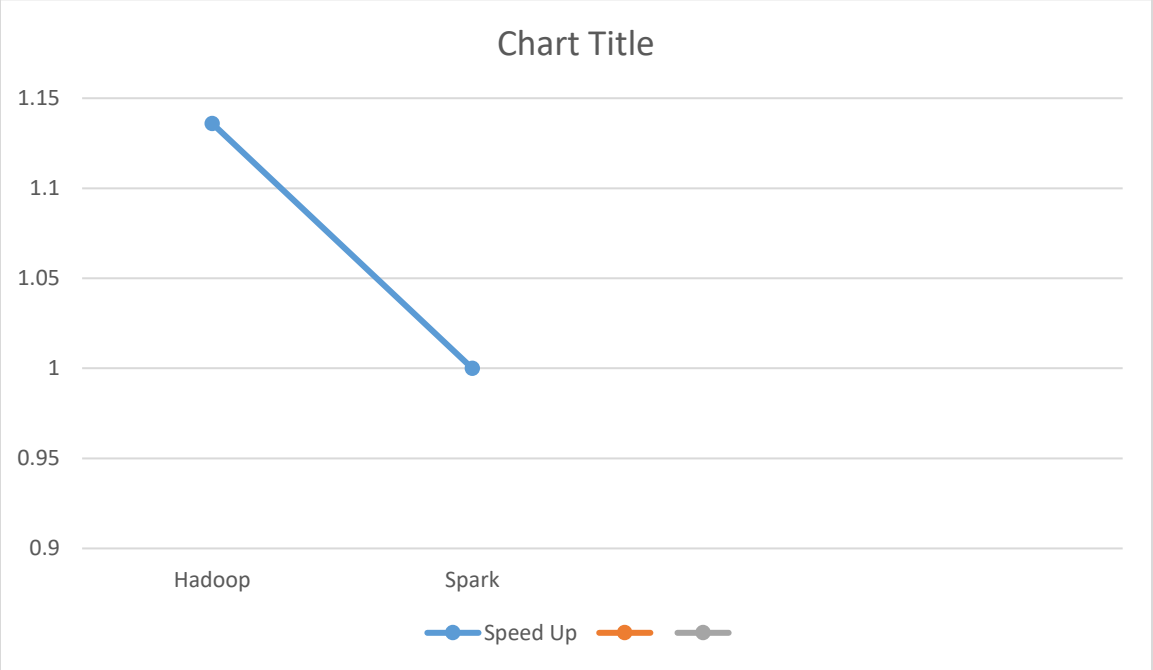
	Speed up	Time
Shared memory	1	1471.159
Hadoop	0.70	1009
Spark	0.545	888



CS553 Programming Assignment #2
Sort on Hadoop/Spark

1 Node 1GB Hadoop Spark

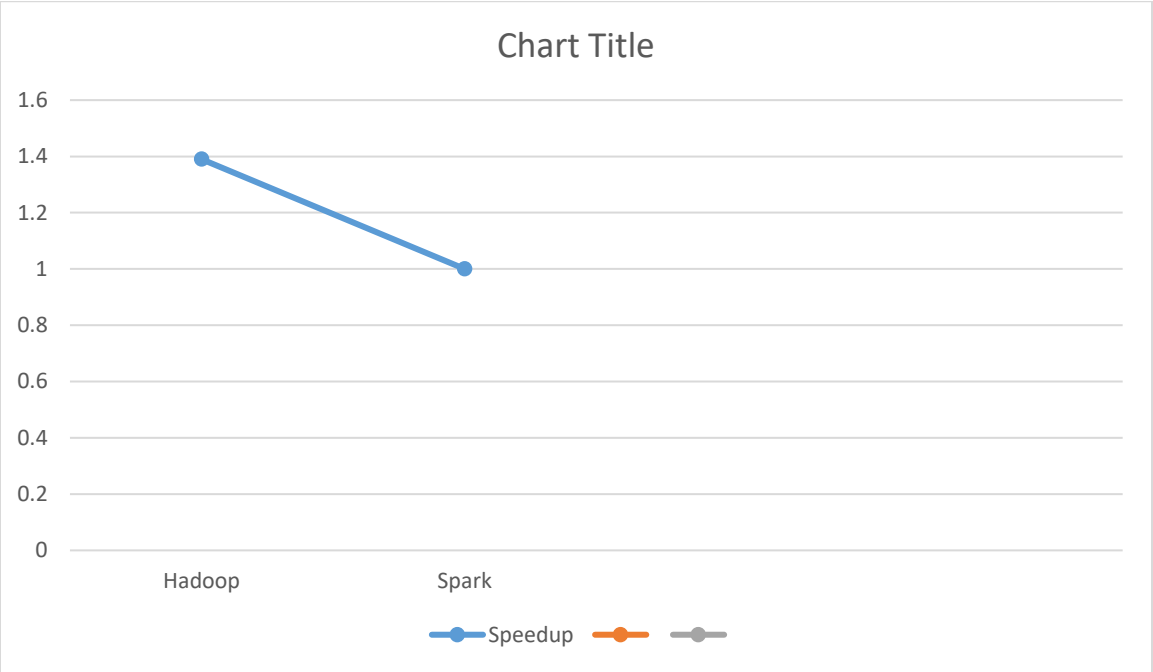
	Speed Up	Time
Hadoop	1.136	1009
Spark	1	888



17 Node 1 GB Hadoop spark.

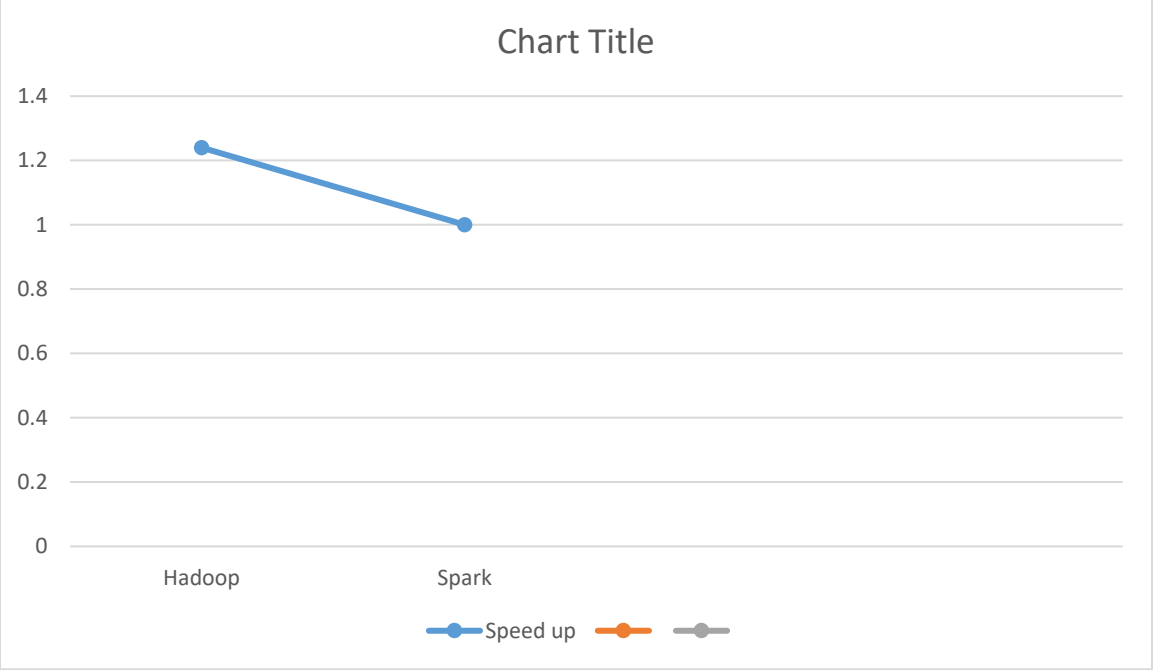
	Speedup	Time
Hadoop	1.39	127
Spark	1	91

CS553 Programming Assignment #2
Sort on Hadoop/Spark



17 Node 100Gb Hadoop Spark speed up

	Speed up	Time
Hadoop	1.24	15600
Spark	1	12600



CS553 Programming Assignment #2
Sort on Hadoop/Spark

Conclusion:

Spark and Shared memory are good at 1 node scale. Spark is good at 16 nodes. Spark is good at 100 and 1000 node scales because of its in-memory primitive data structure RDD (Resilient distributed dataset).

Comparison with cloudsort

2014 winners

Hadoop- 102.5 TB in 4,328 seconds
Throughput: 23682.99 mb/sec
2100 nodes (2 2.3Ghz hexcore Xeon E5-2630, 64 GB memory, 12x3TB disks)

Spark- 100 TB in 1,406 seconds
Throughput: 71123.75 mb/sec
207 Amazon EC2 i2.8xlarge nodes x
(32 vCores - 2.5Ghz Intel Xeon E5-2670 v2, 244GB memory, 8x800 GB SSD)

2013 Winners

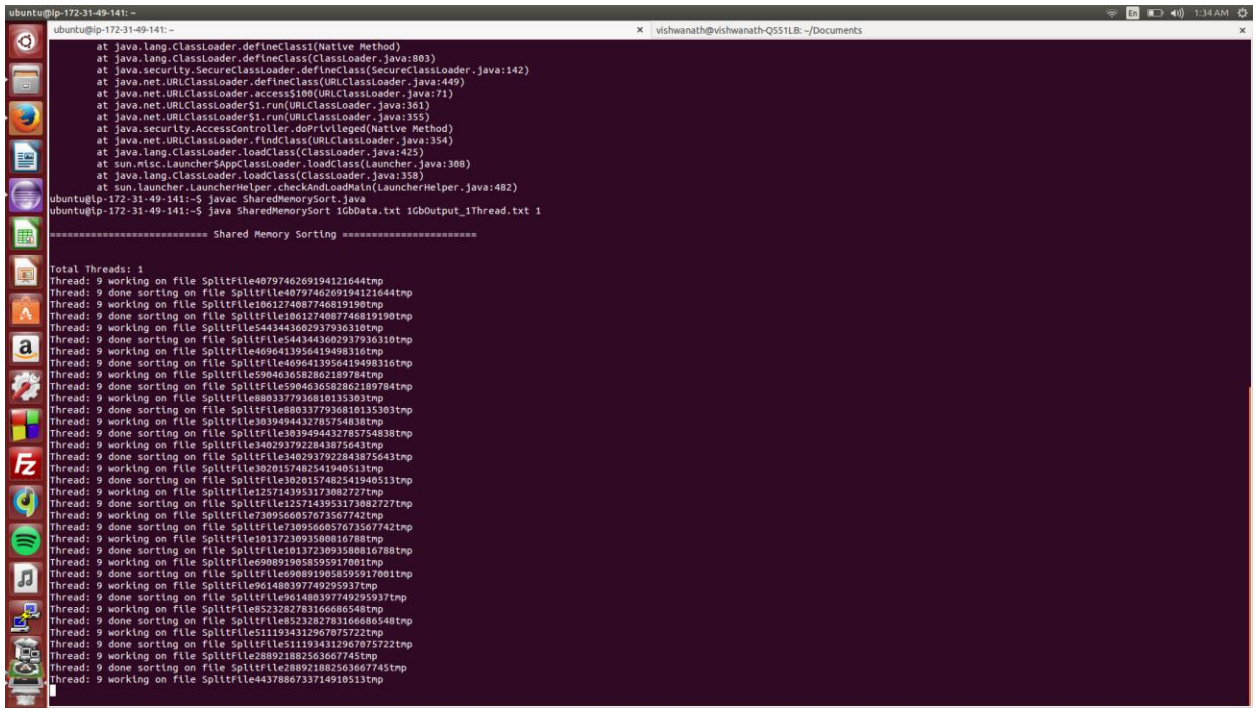
Hadoop: 102.5 TB in 4,328 seconds
Throughput: 23682.99 mb/sec
2100 nodes x
(2 2.3Ghz hexcore Xeon E5-2630, 64 GB memory, 12x3TB disks)

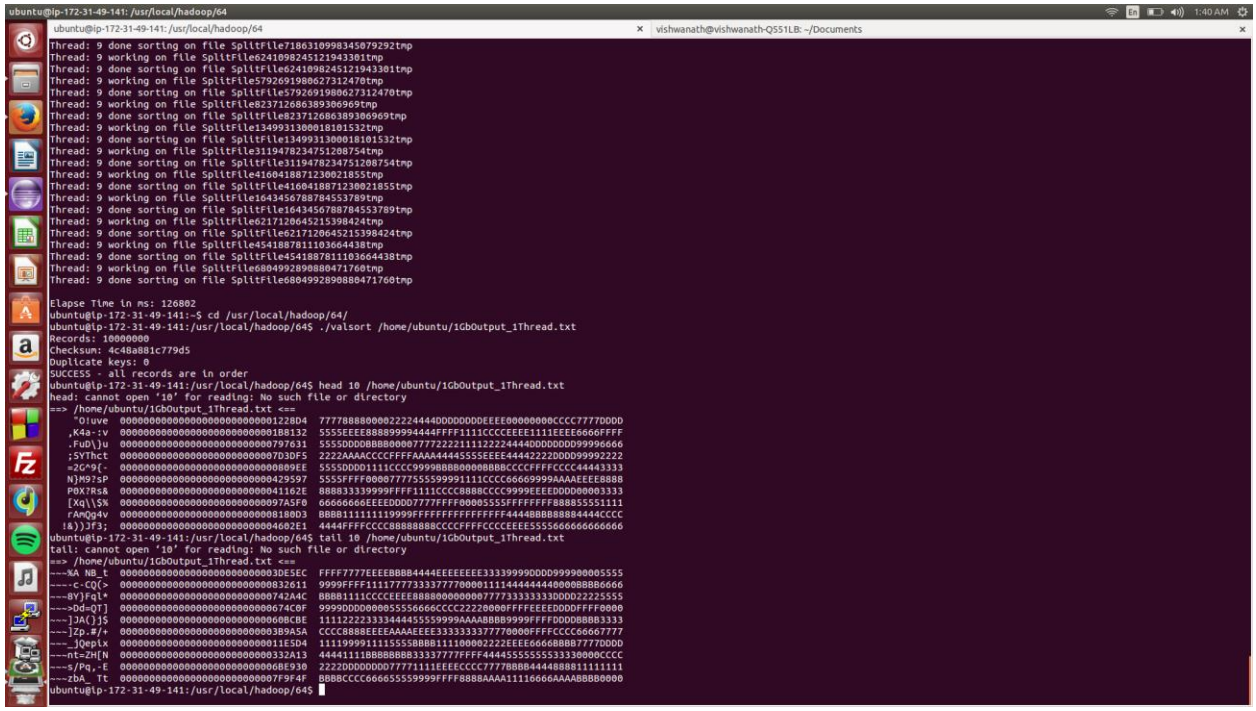
2013 and 2014 winners have more throughput than our experiment throughput. Because of system specifications .i.e. amount of memory, Disk type SSD. Configurations settings.

Screen Shots

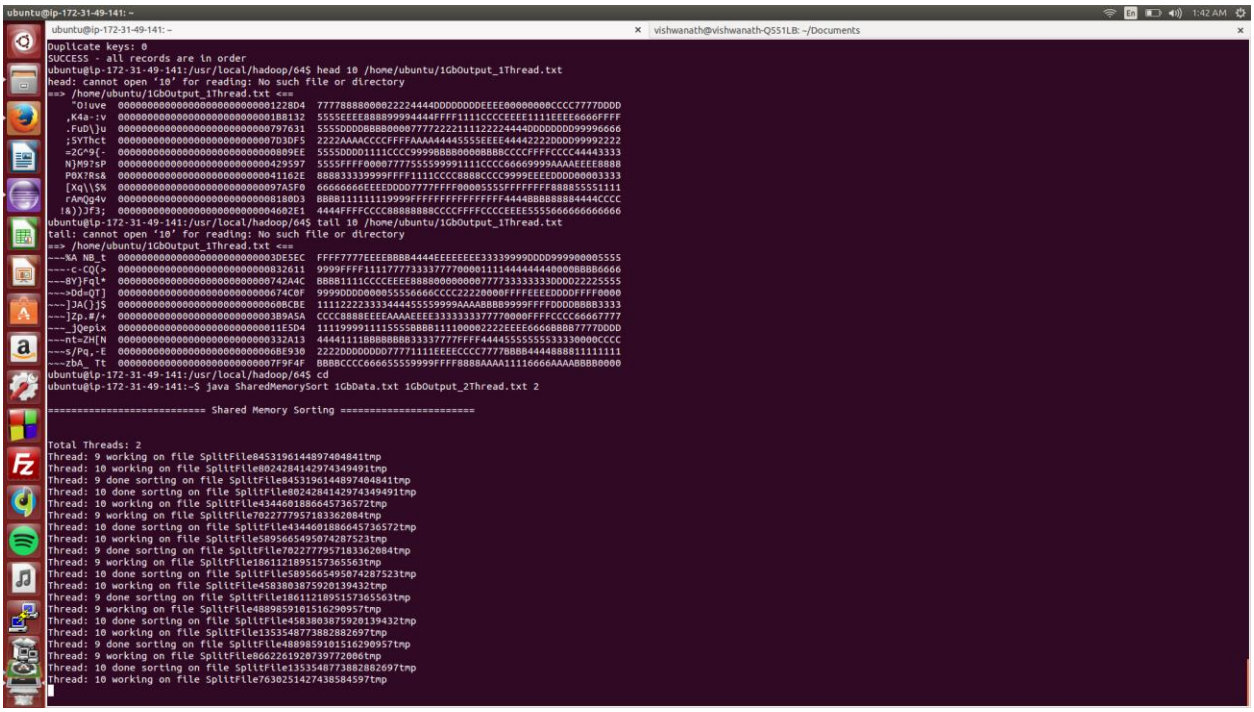
CS553 Programming Assignment #2
Sort on Hadoop/Spark

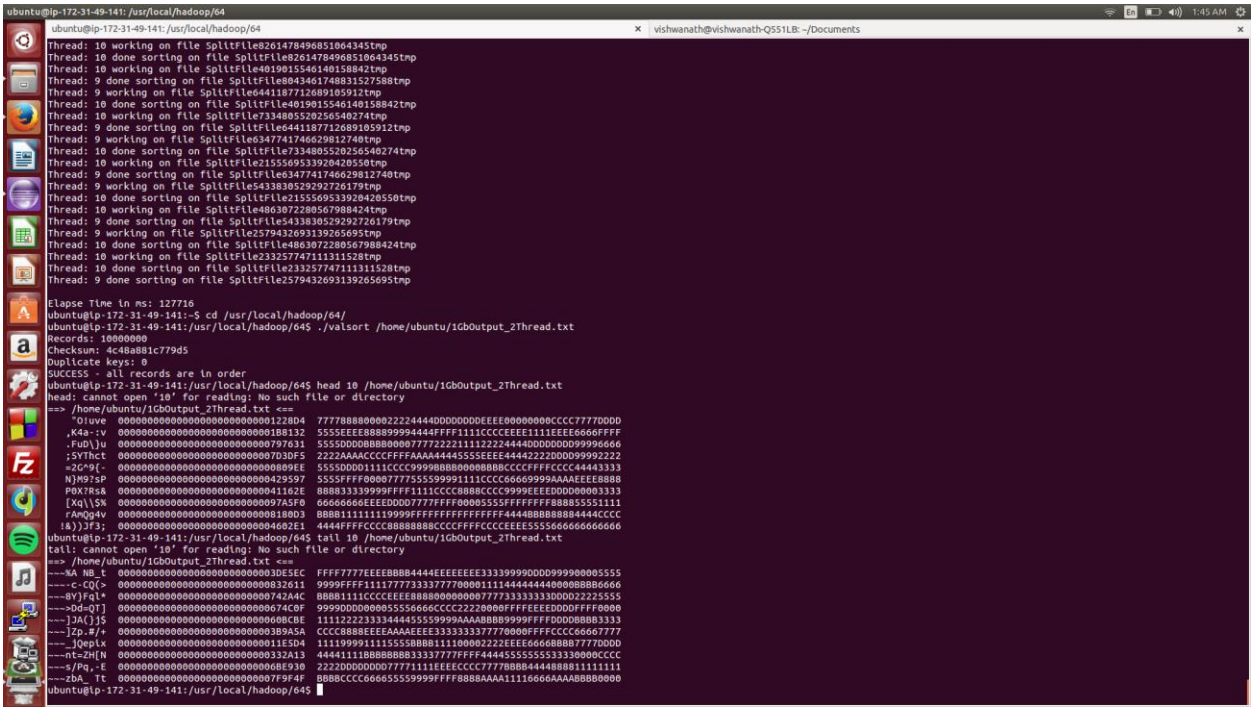
1. Shared memory



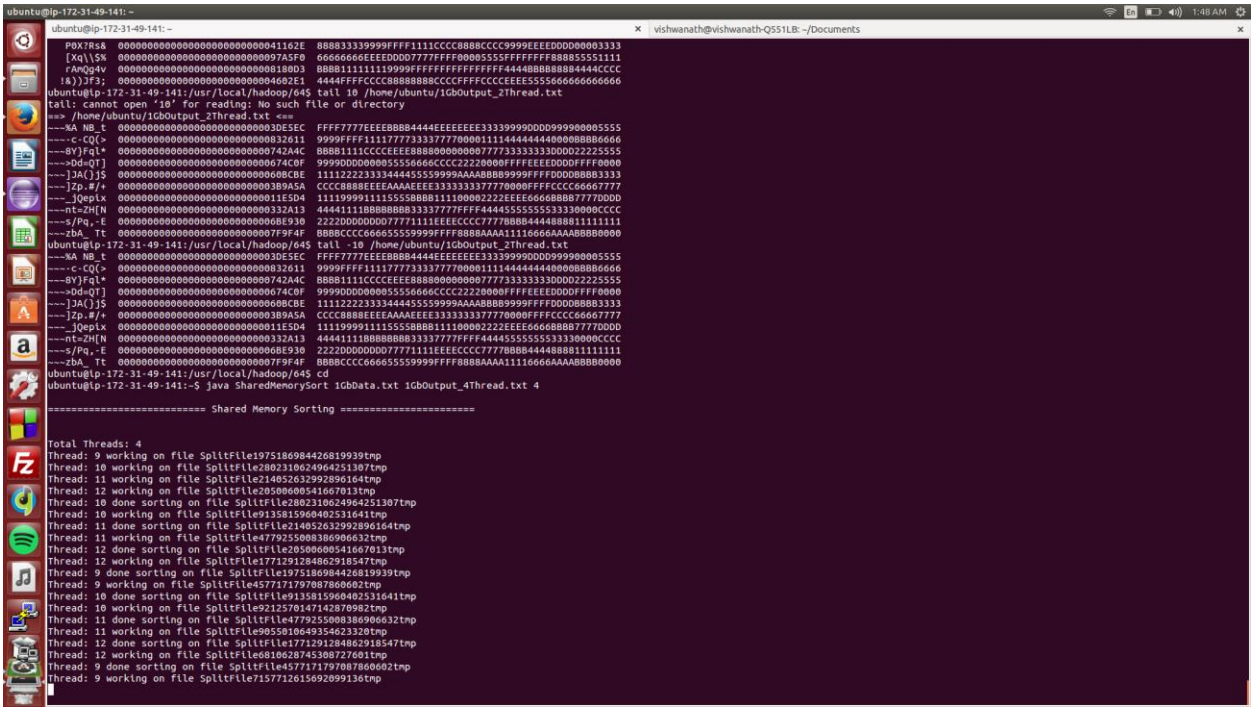


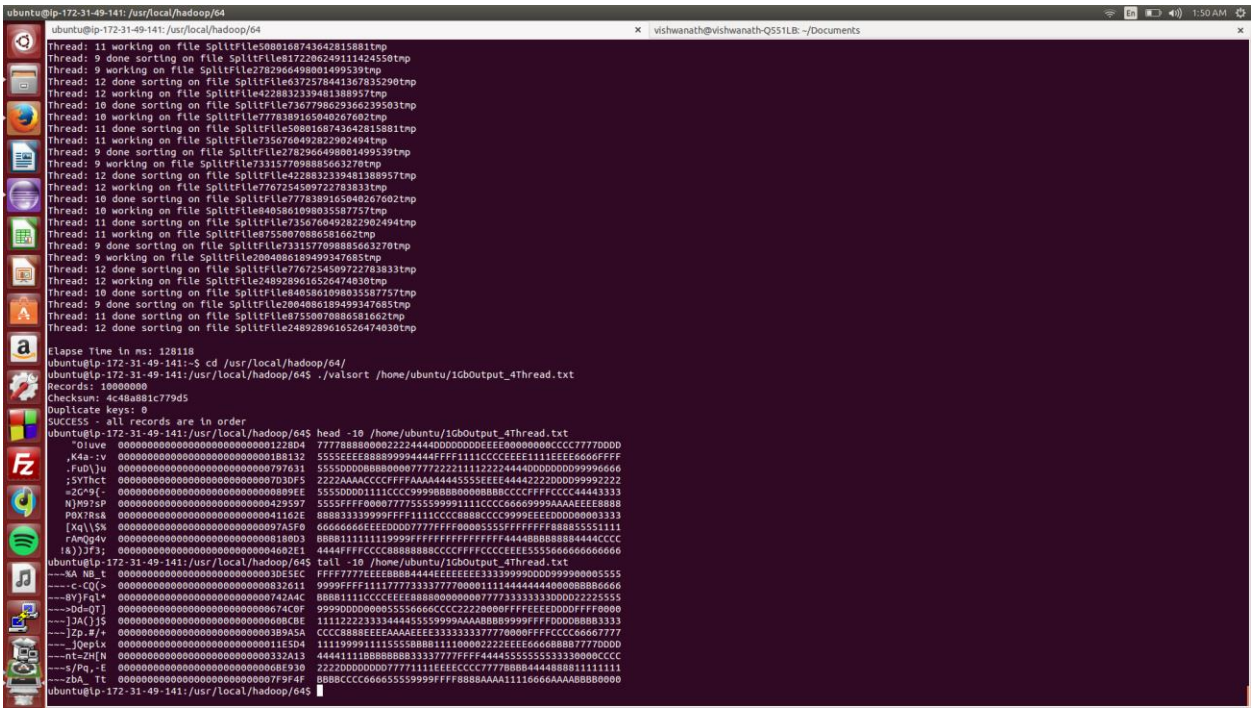
CS553 Programming Assignment #2
Sort on Hadoop/Spark



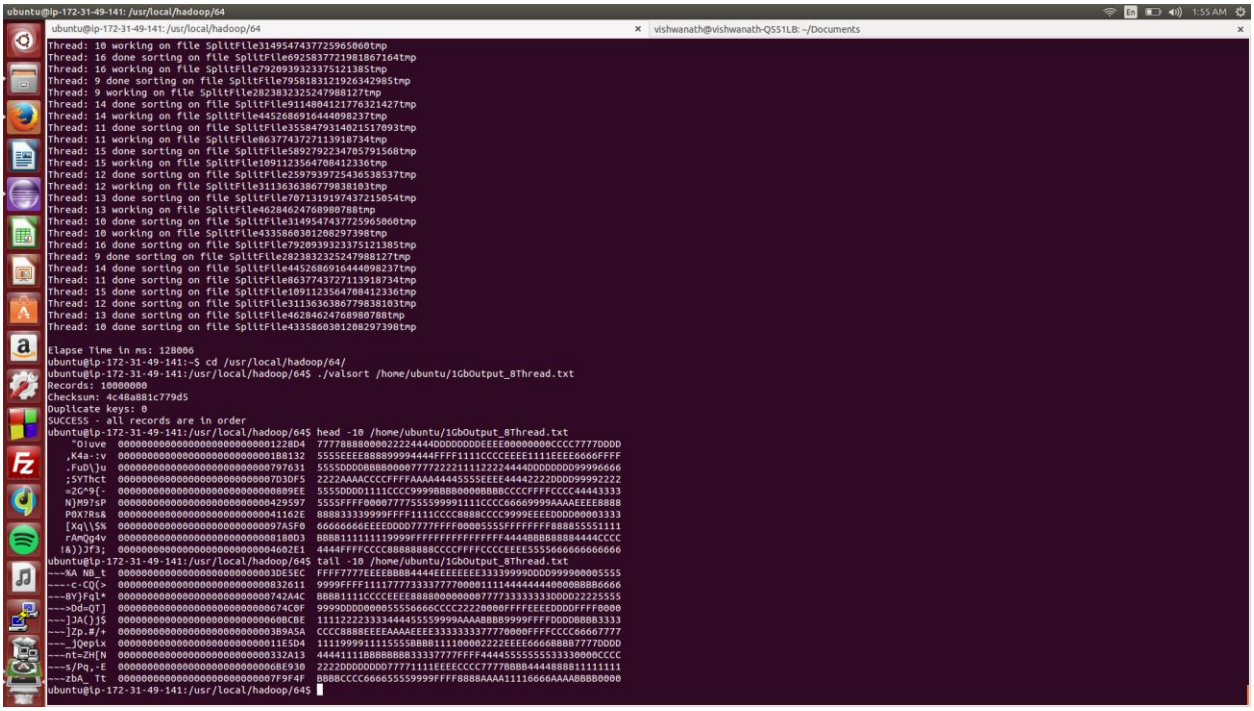


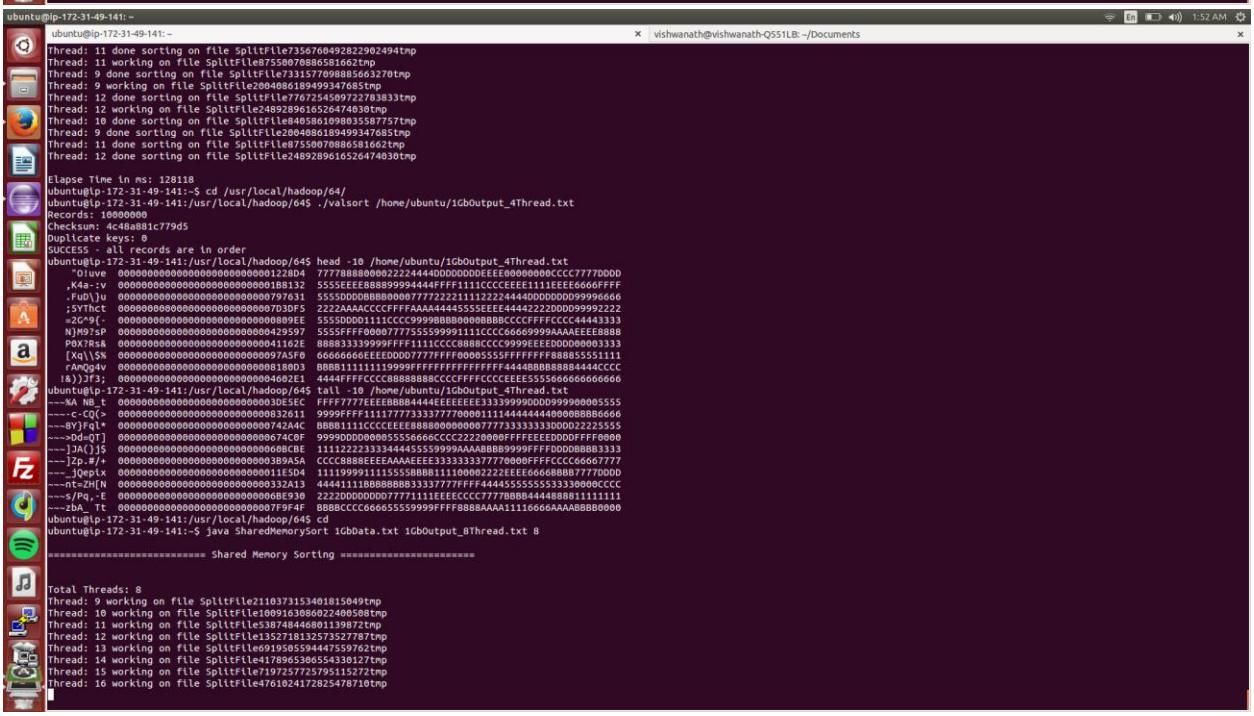
CS553 Programming Assignment #2
Sort on Hadoop/Spark



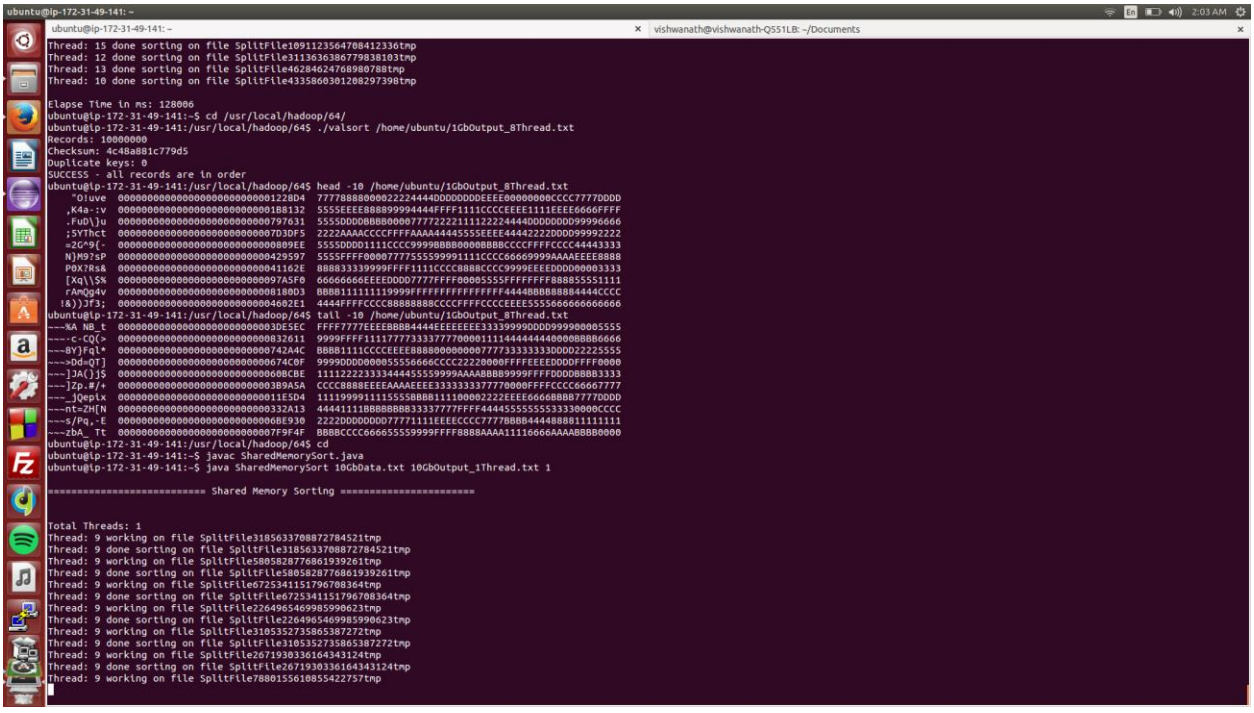


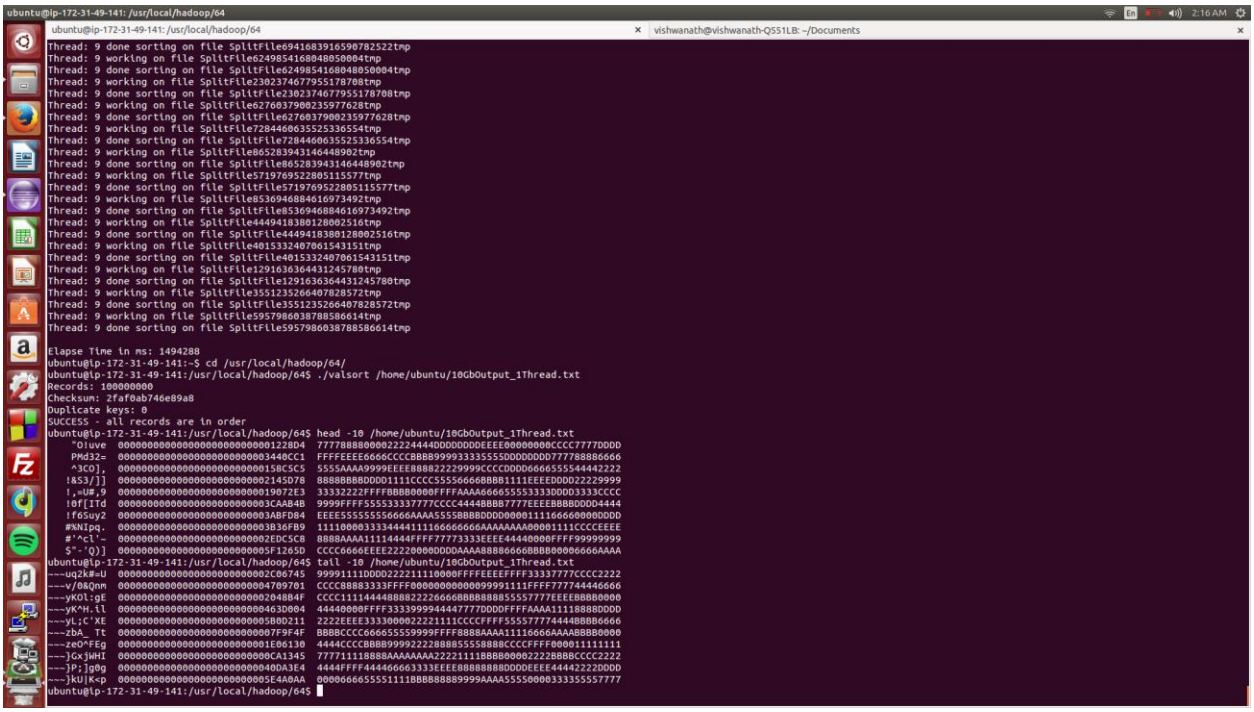
CS553 Programming Assignment #2
Sort on Hadoop/Spark



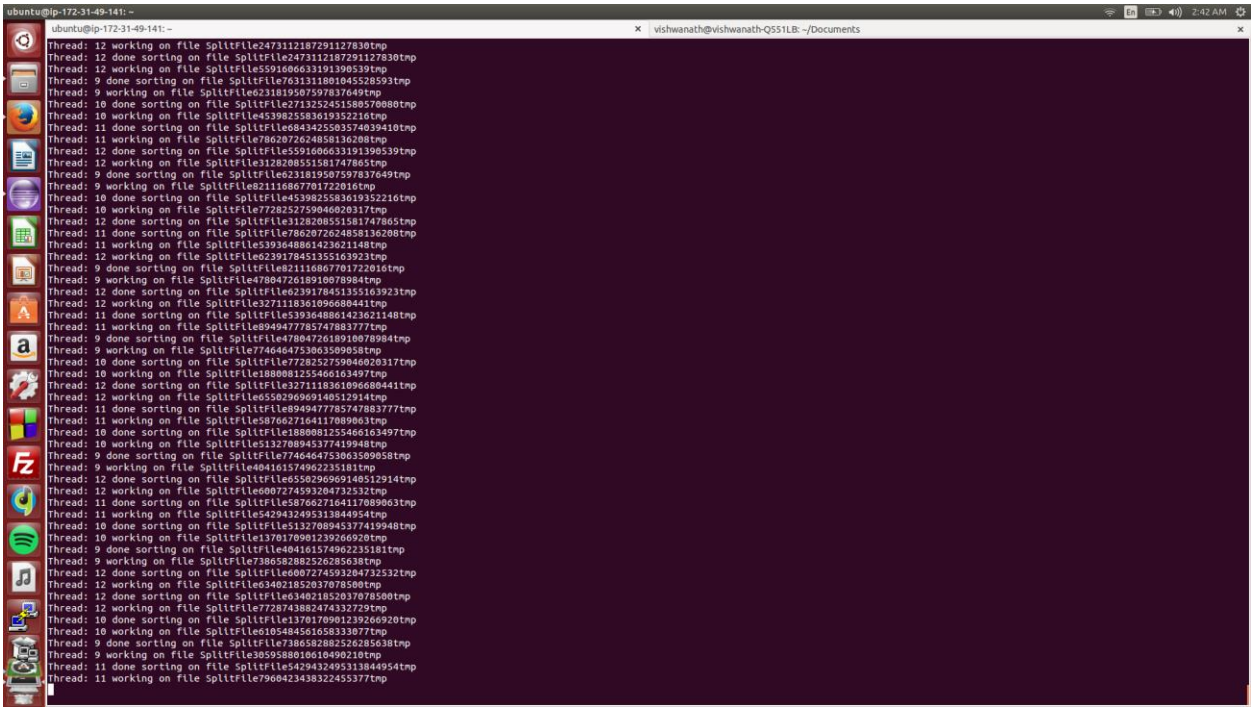


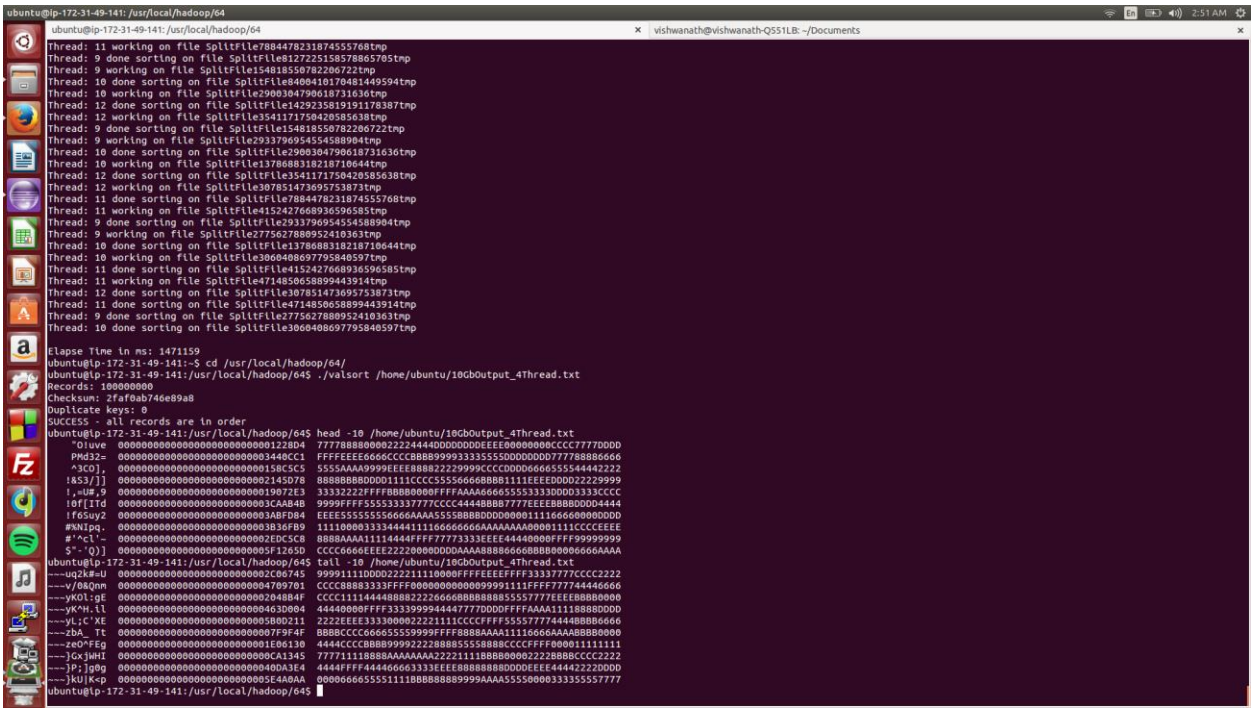
CS553 Programming Assignment #2
Sort on Hadoop/Spark



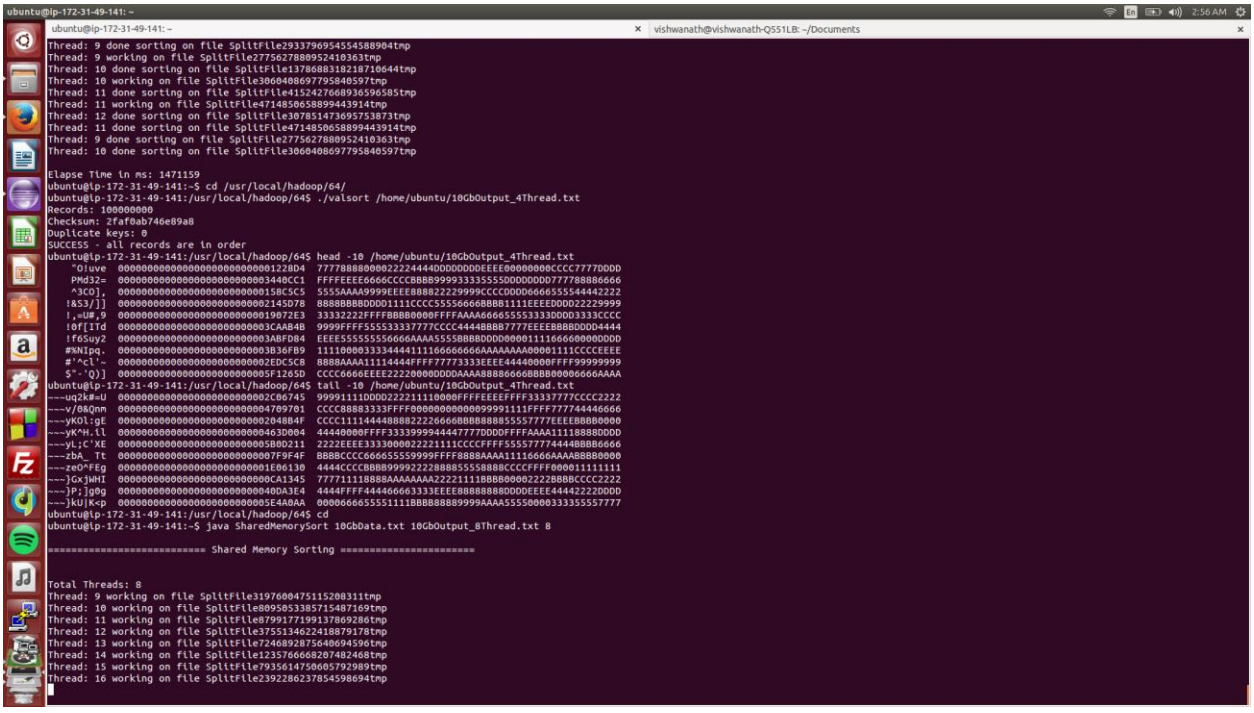


CS553 Programming Assignment #2
Sort on Hadoop/Spark

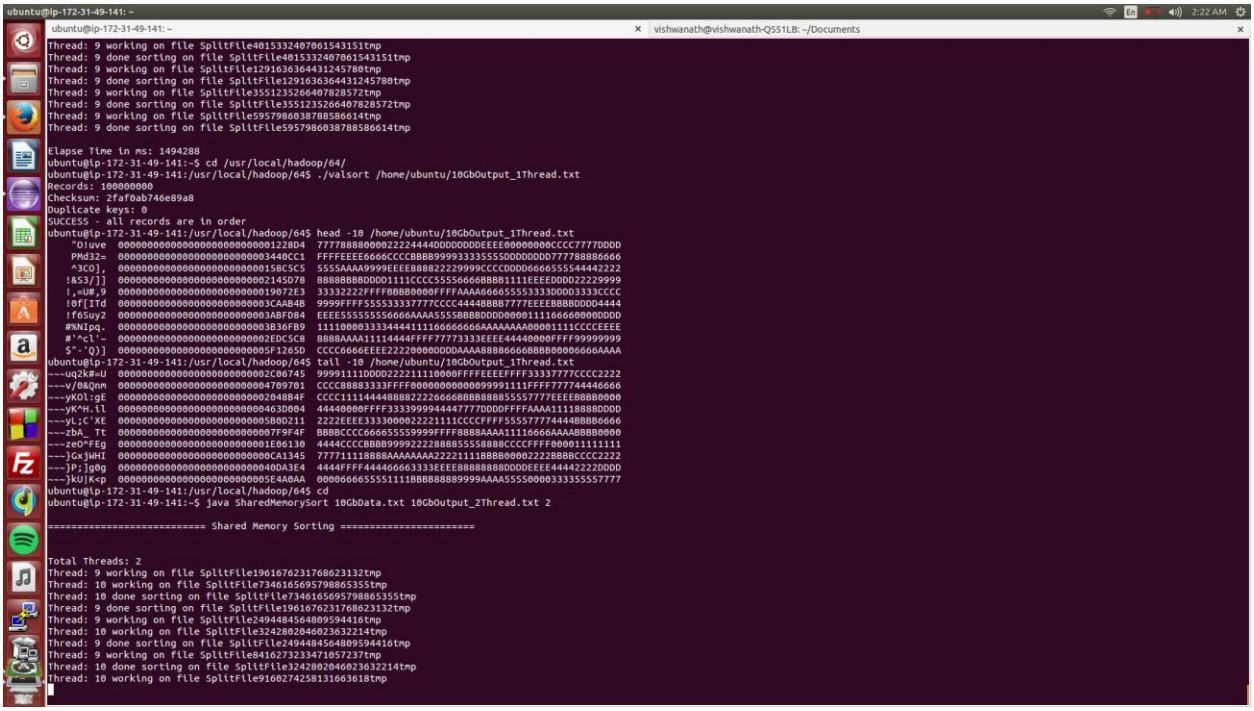


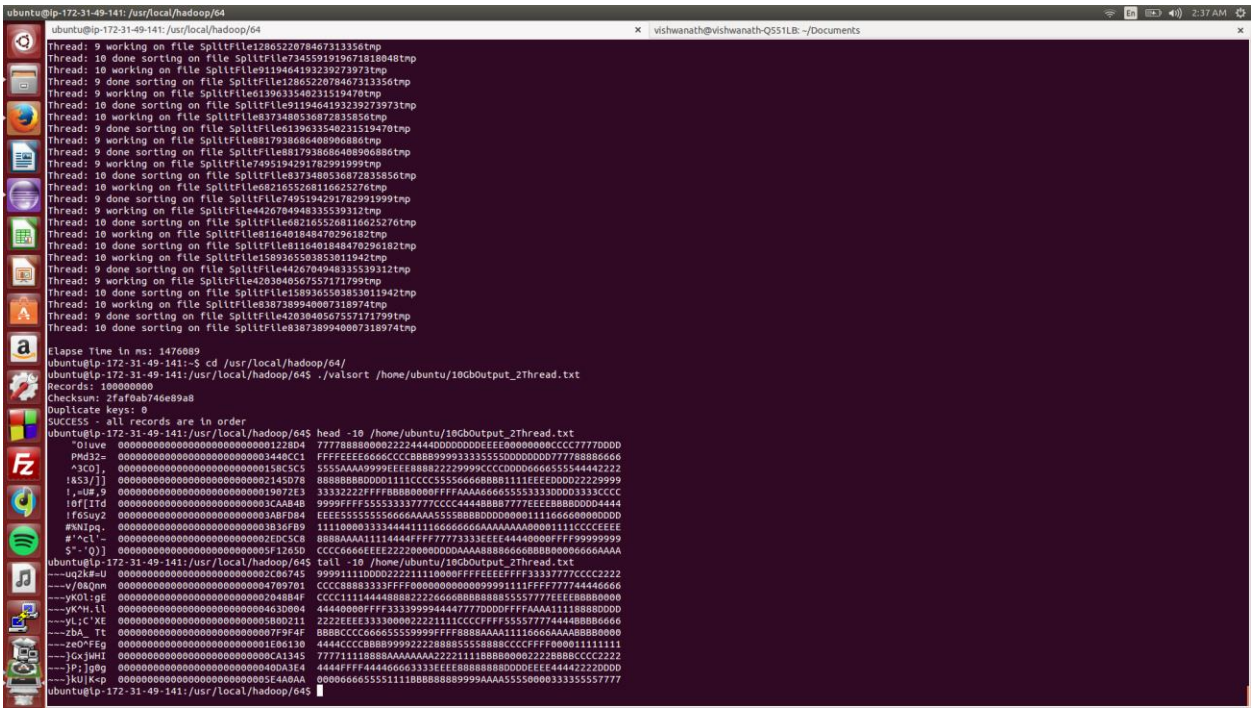


CS553 Programming Assignment #2
Sort on Hadoop/Spark

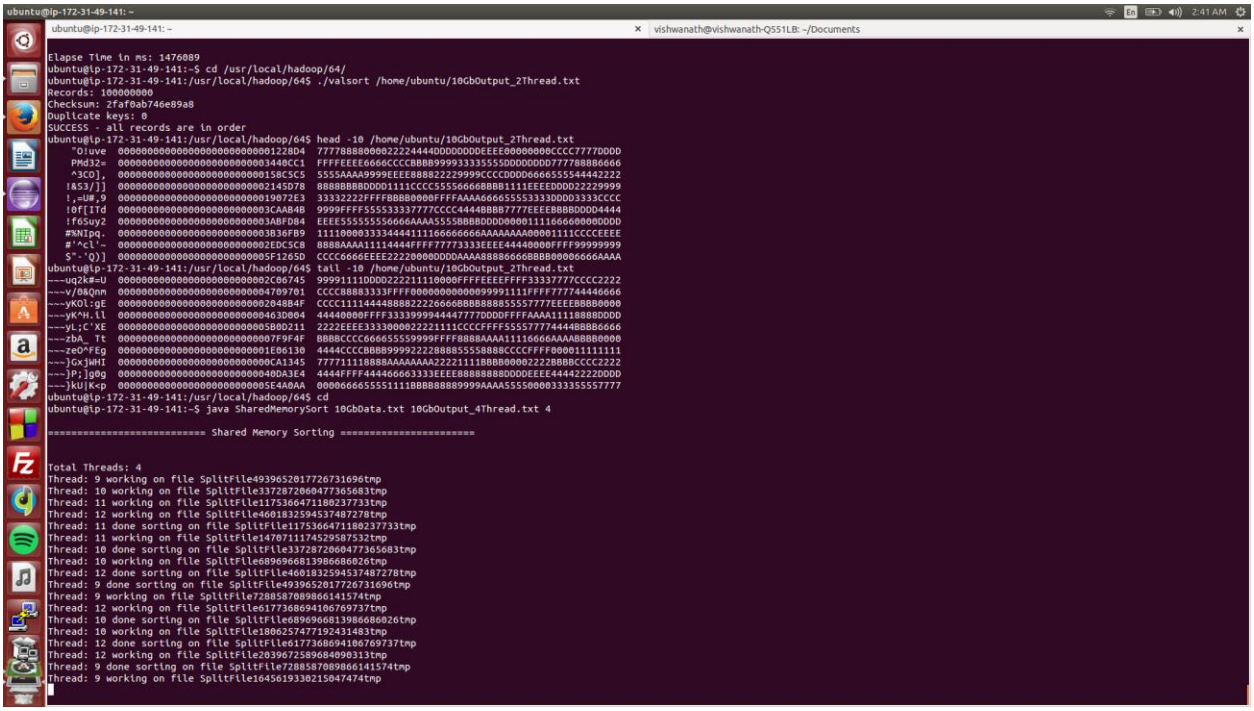


CS553 Programming Assignment #2
Sort on Hadoop/Spark

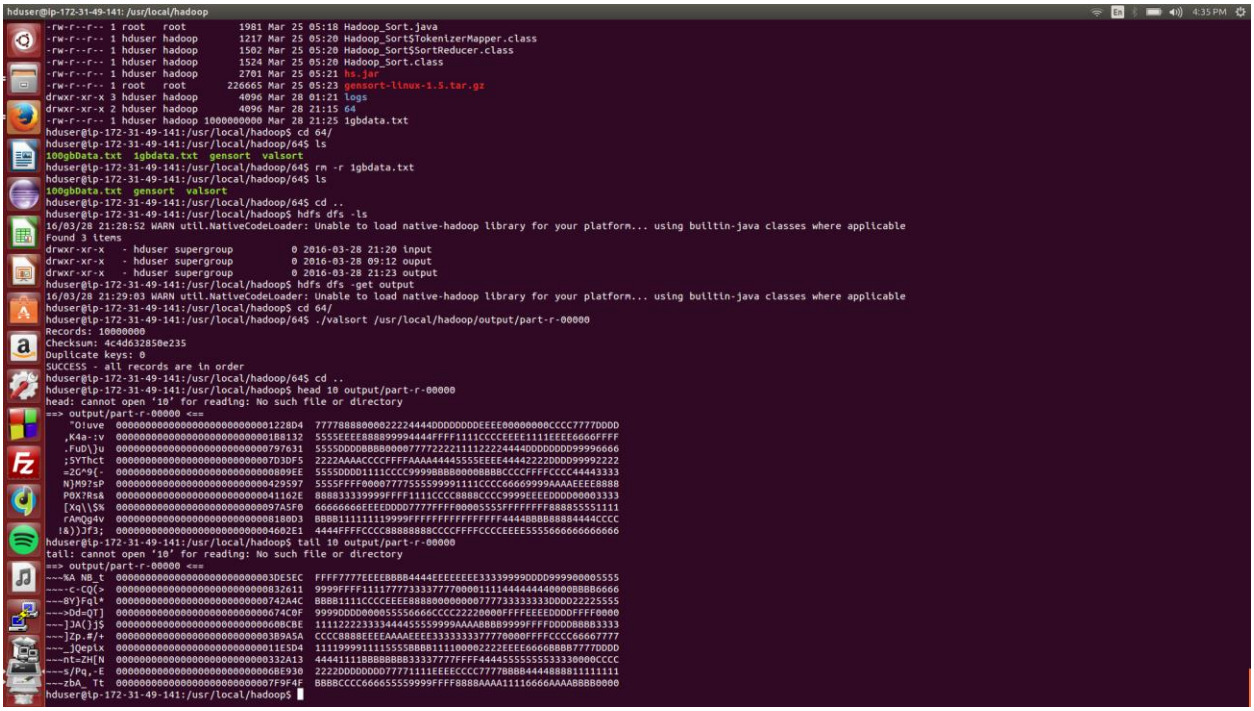




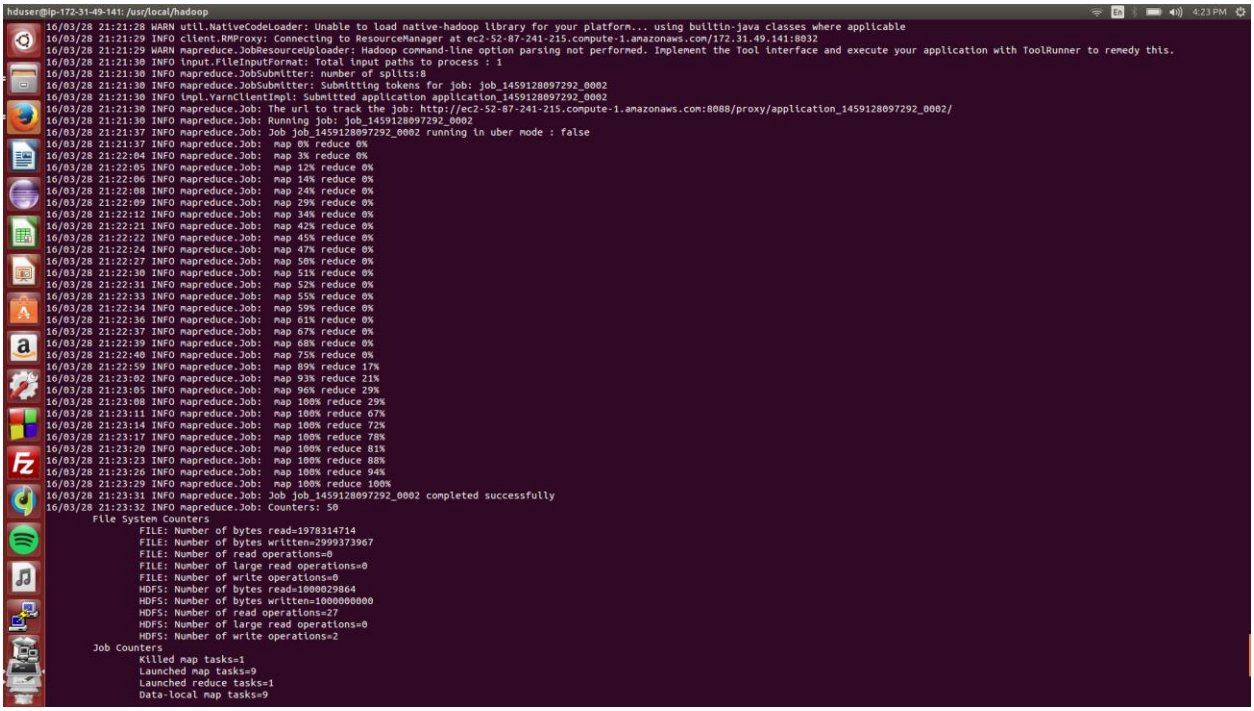
CS553 Programming Assignment #2
Sort on Hadoop/Spark

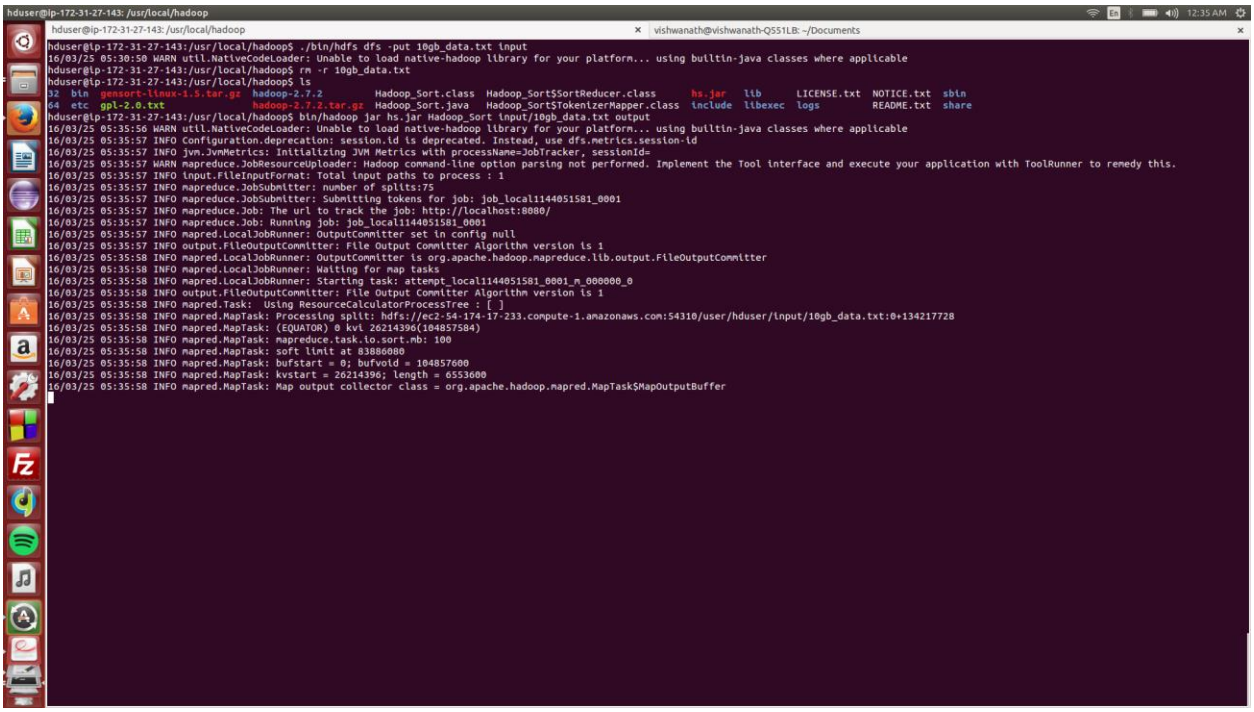


2. Hadoop

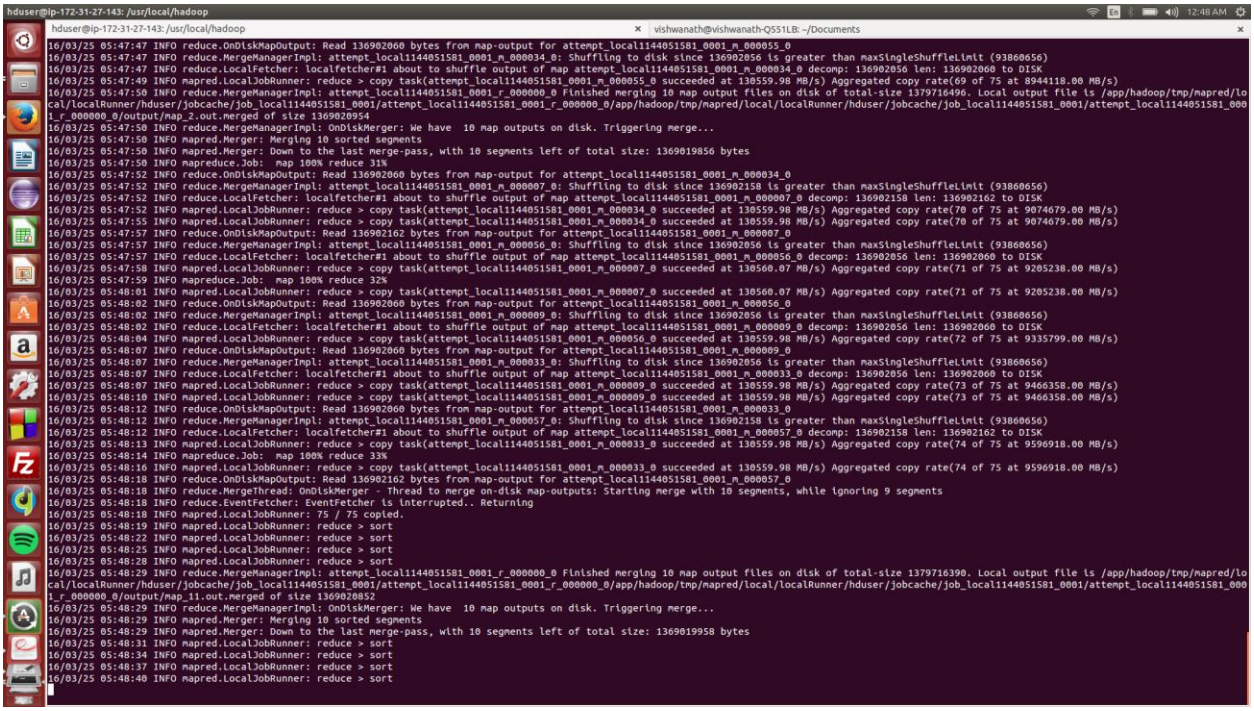


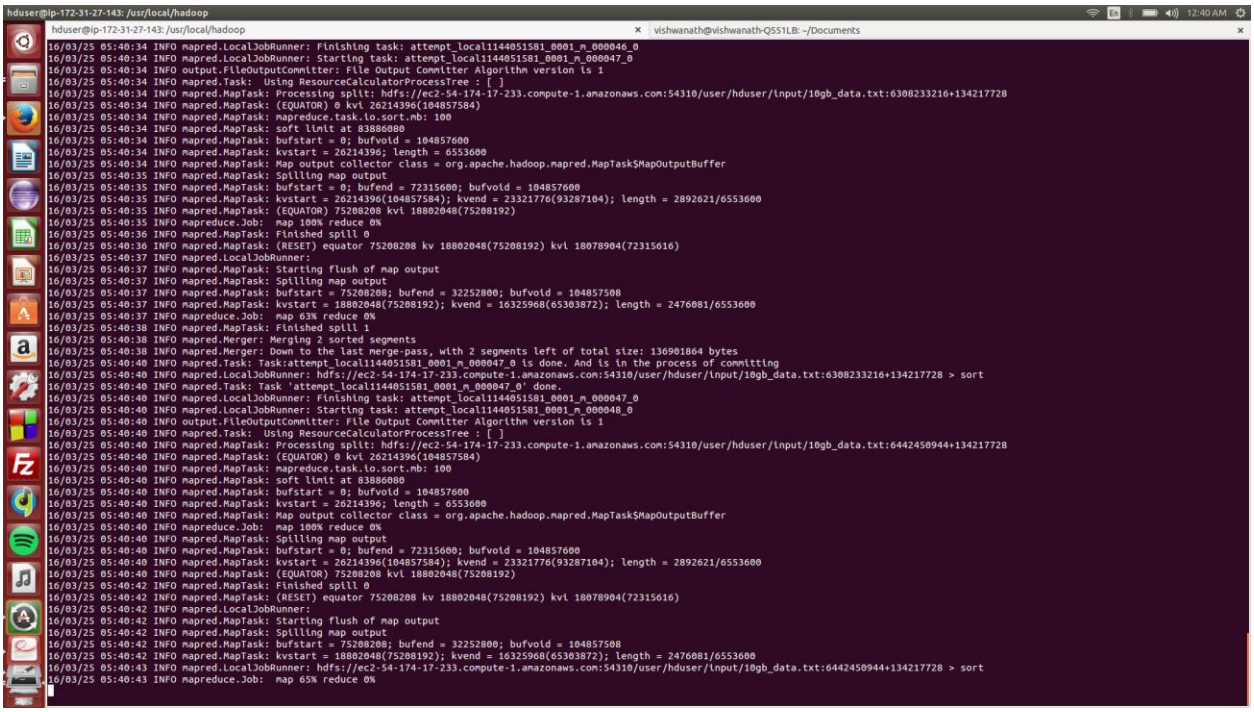
CS553 Programming Assignment #2
Sort on Hadoop/Spark



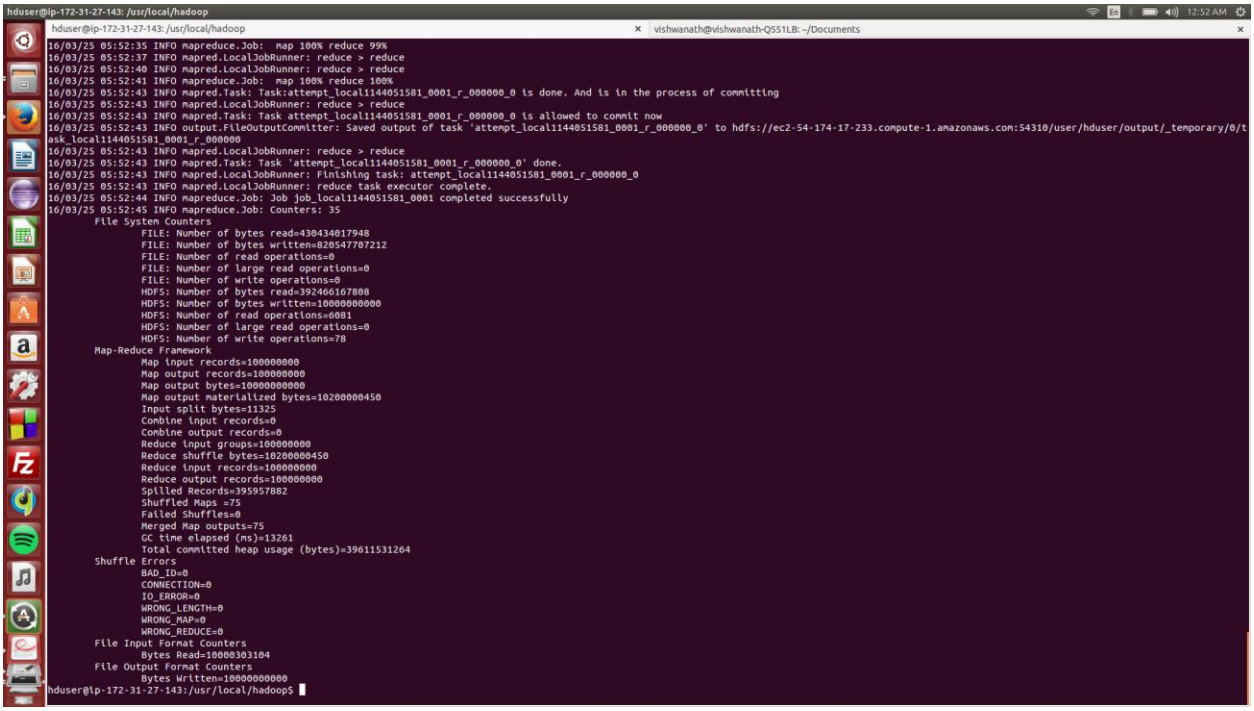


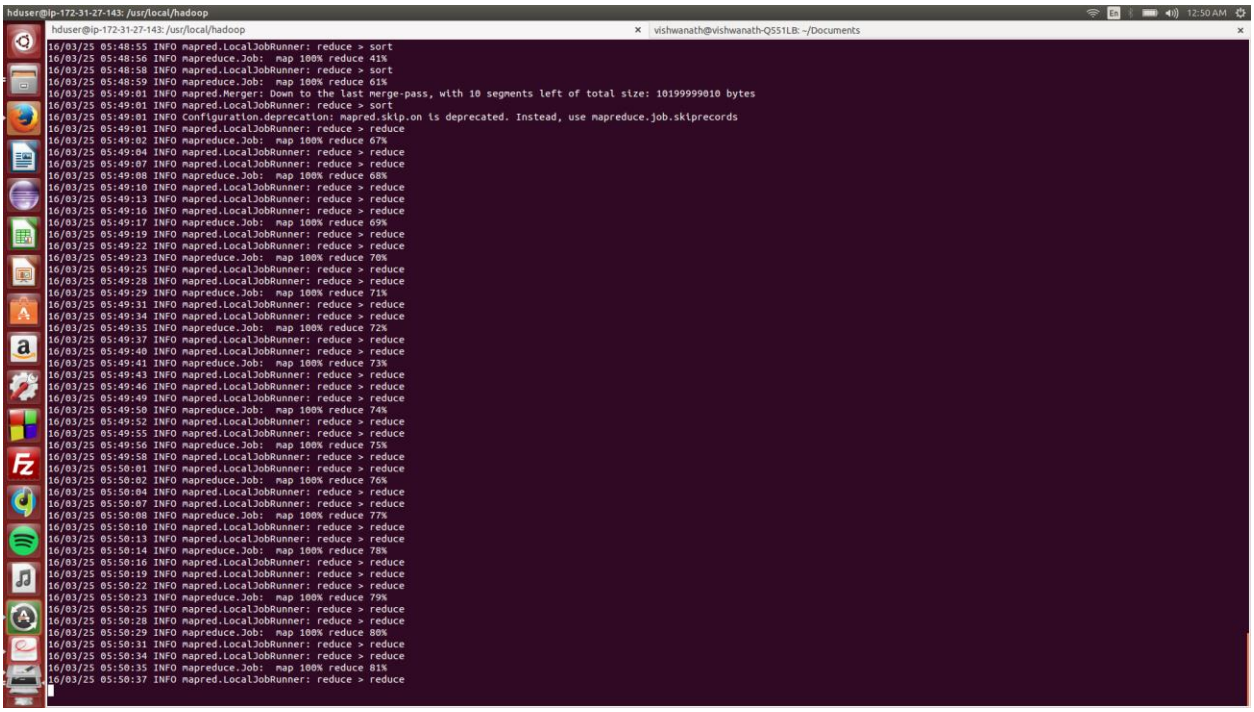
CS553 Programming Assignment #2
Sort on Hadoop/Spark



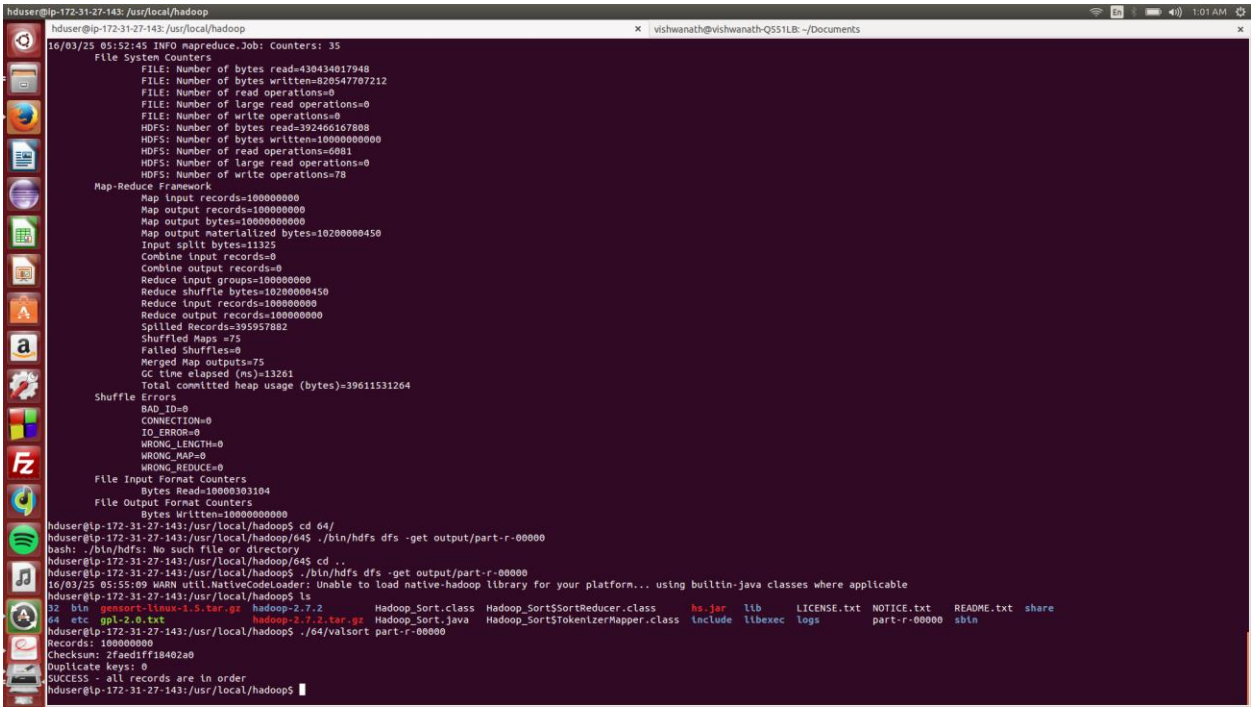


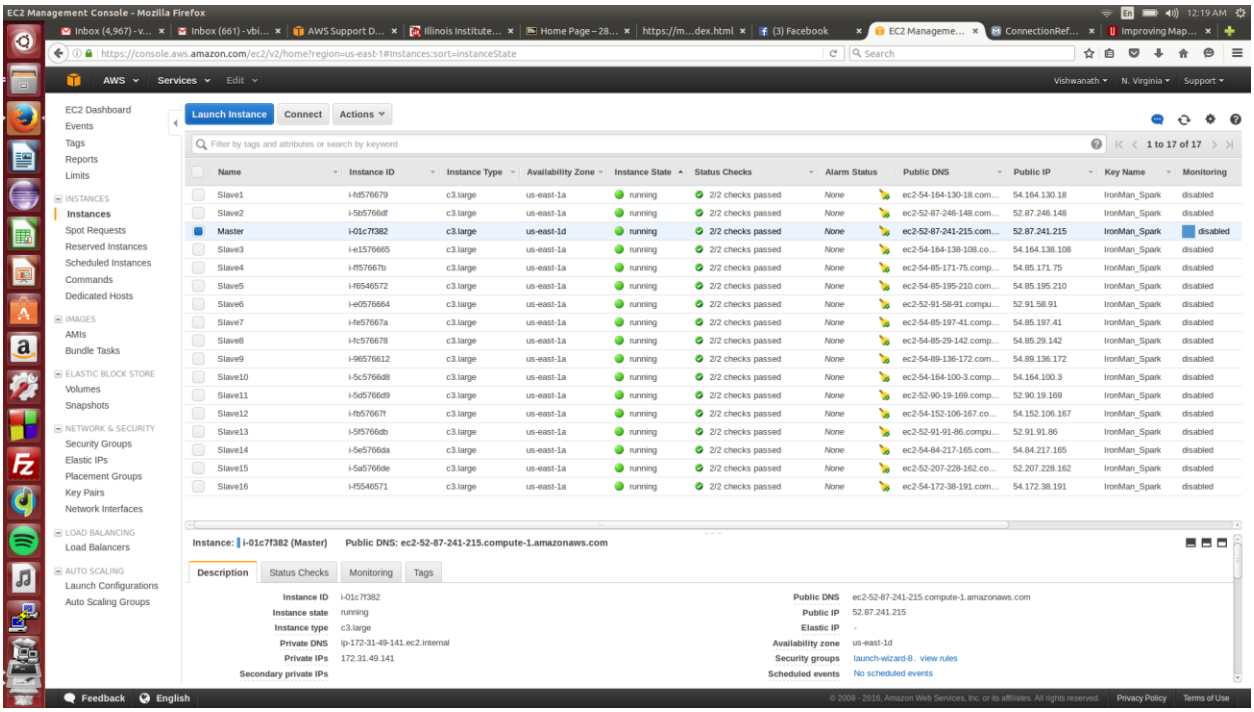
CS553 Programming Assignment #2
Sort on Hadoop/Spark



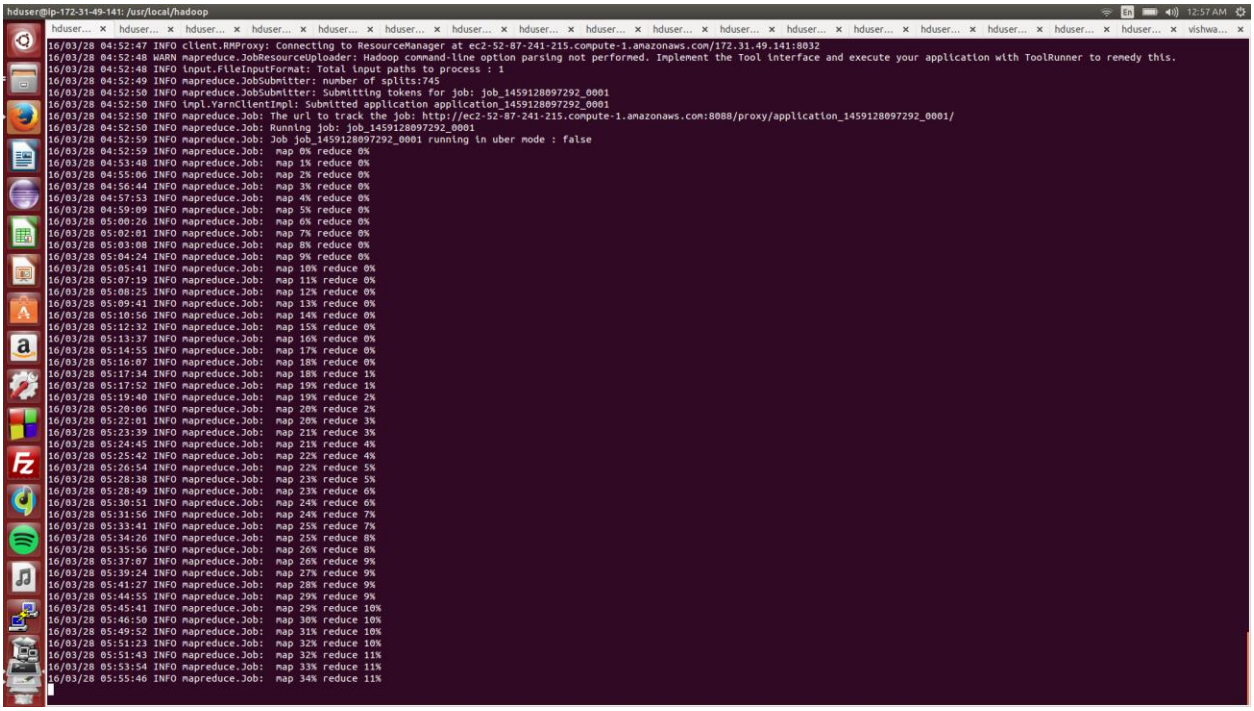


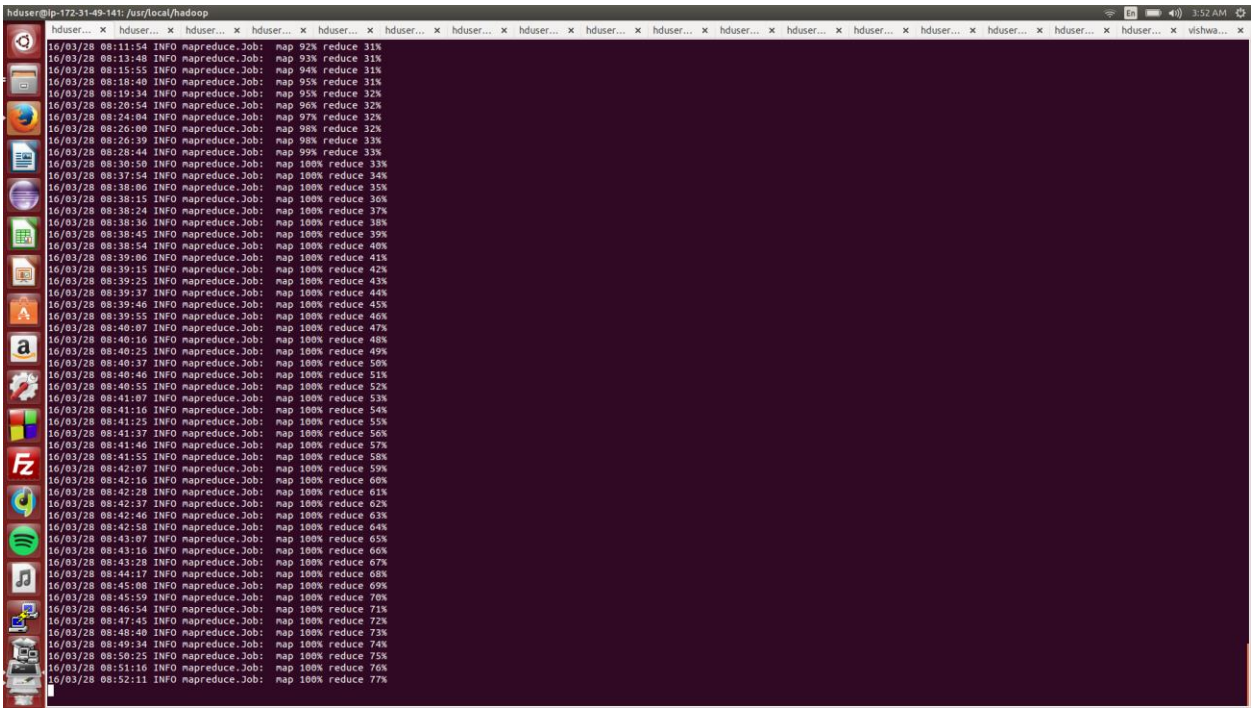
CS553 Programming Assignment #2
Sort on Hadoop/Spark



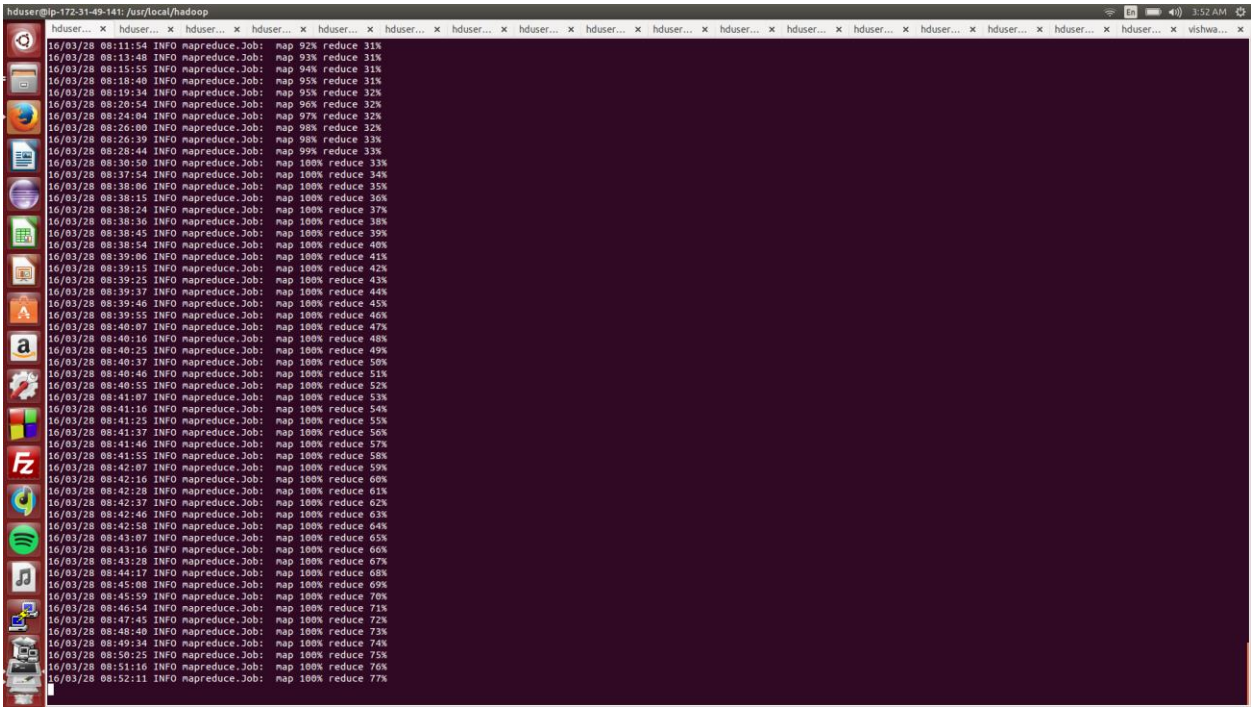


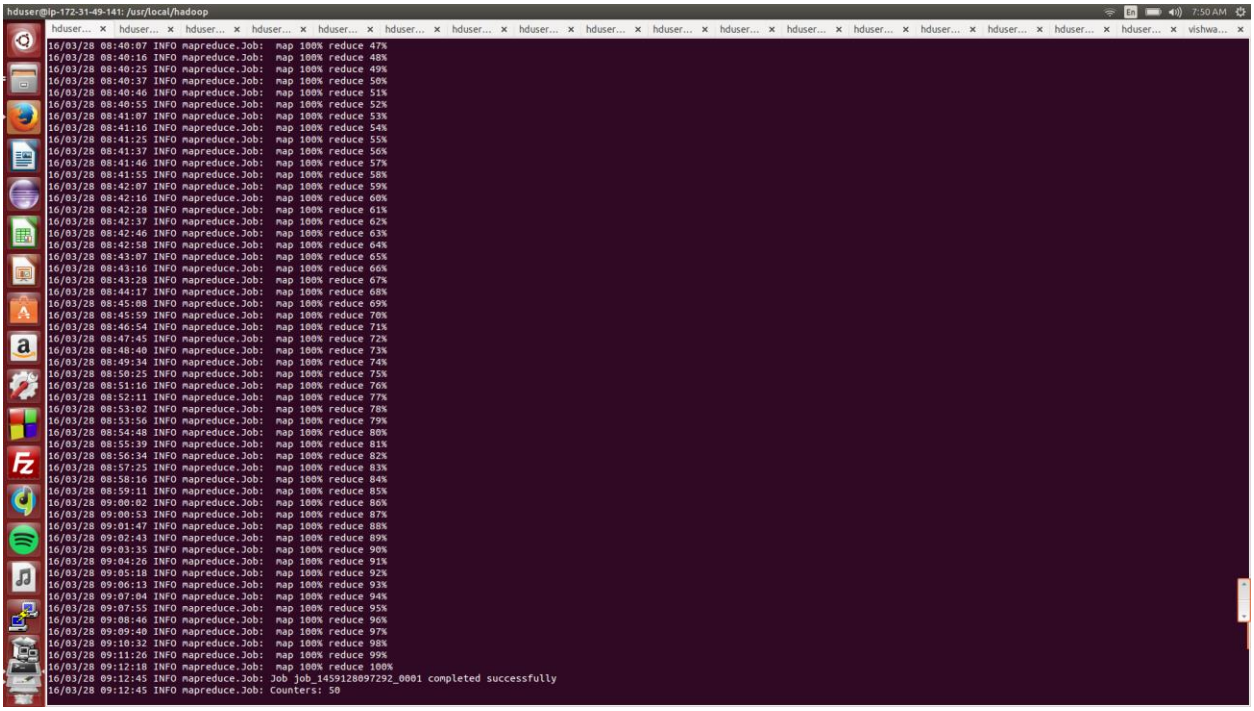
CS553 Programming Assignment #2
Sort on Hadoop/Spark



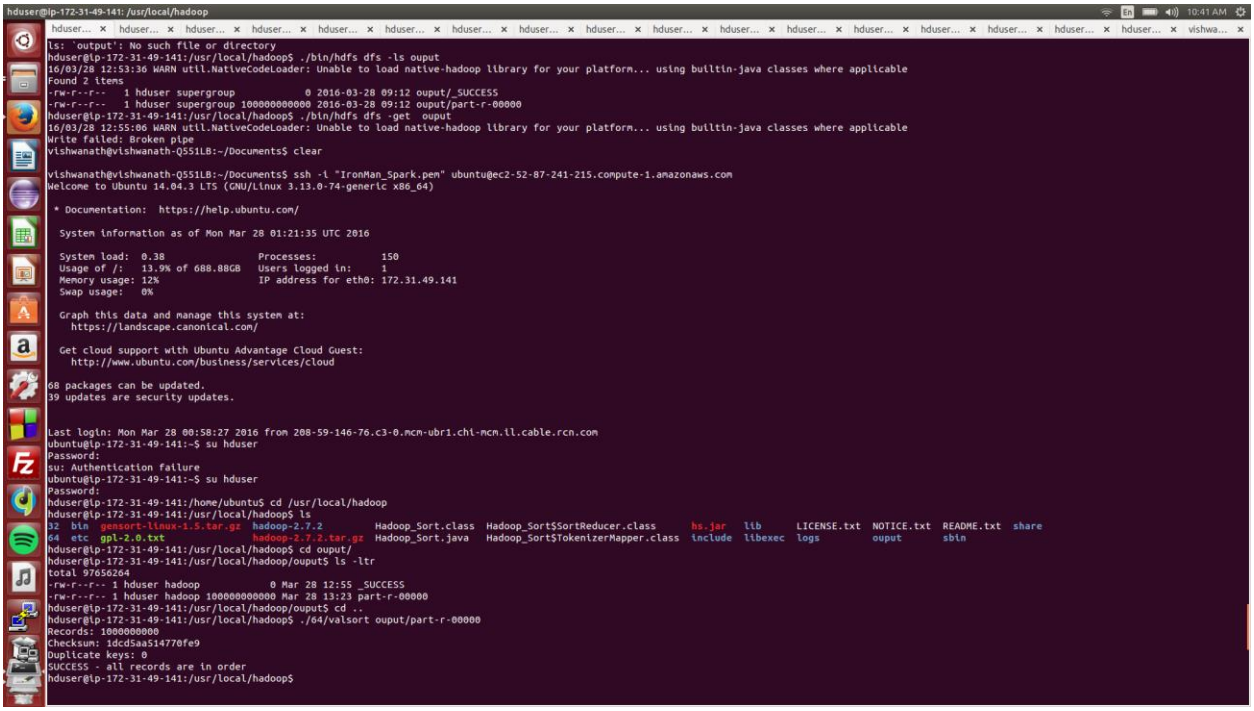


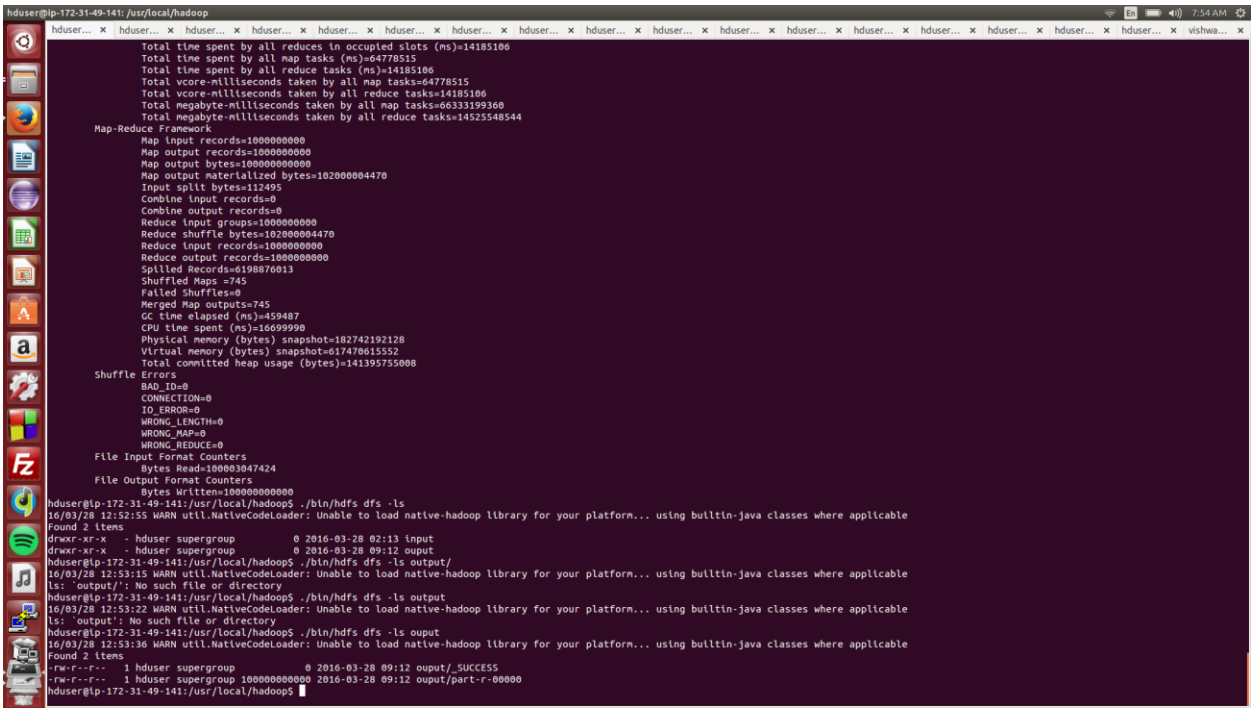
CS553 Programming Assignment #2
Sort on Hadoop/Spark



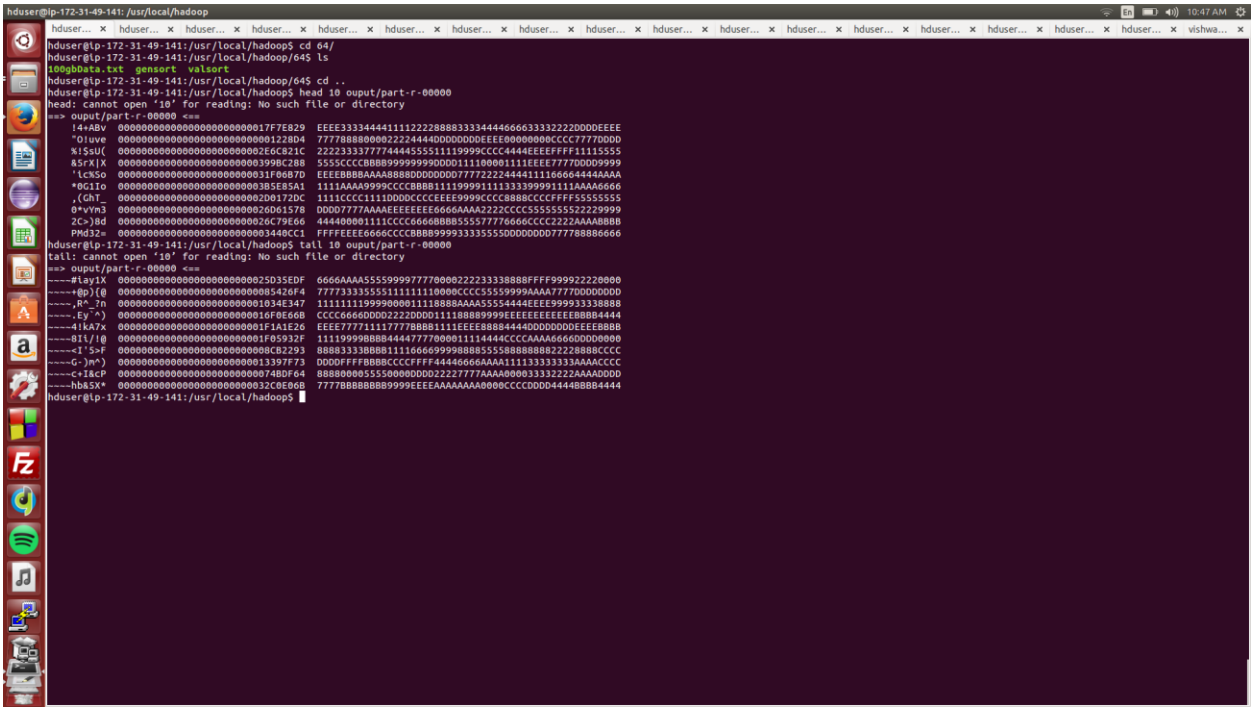


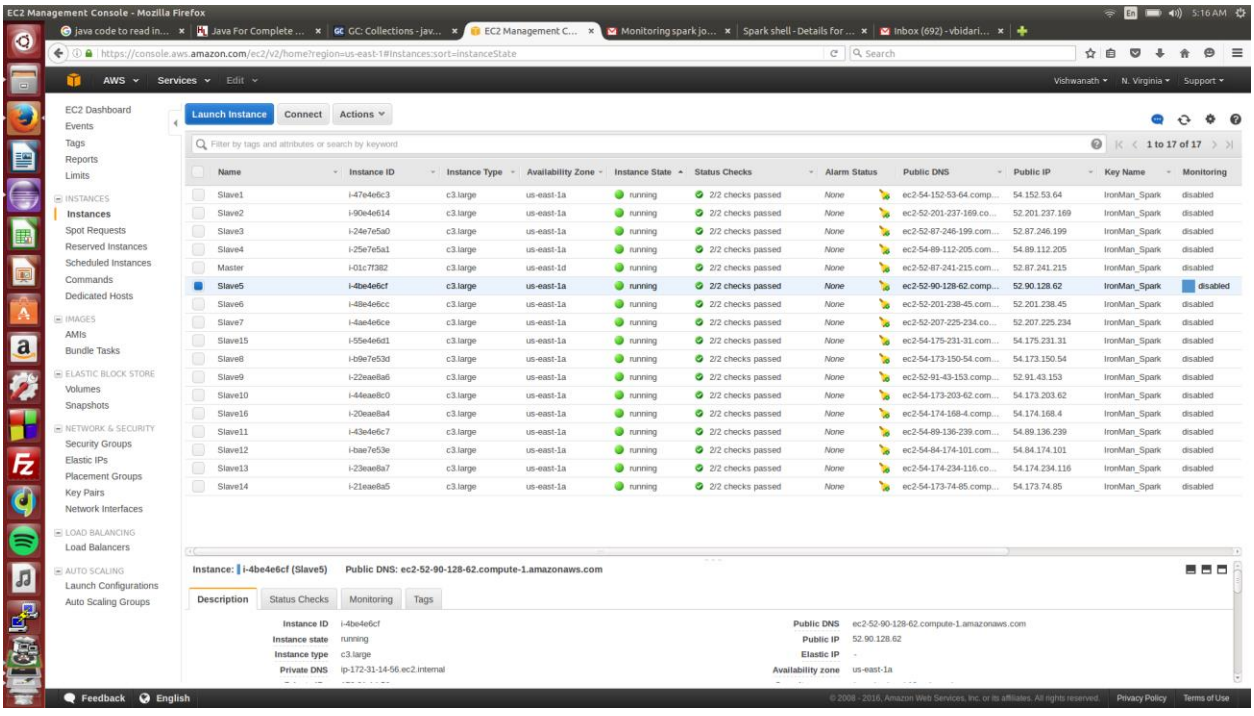
CS553 Programming Assignment #2
Sort on Hadoop/Spark



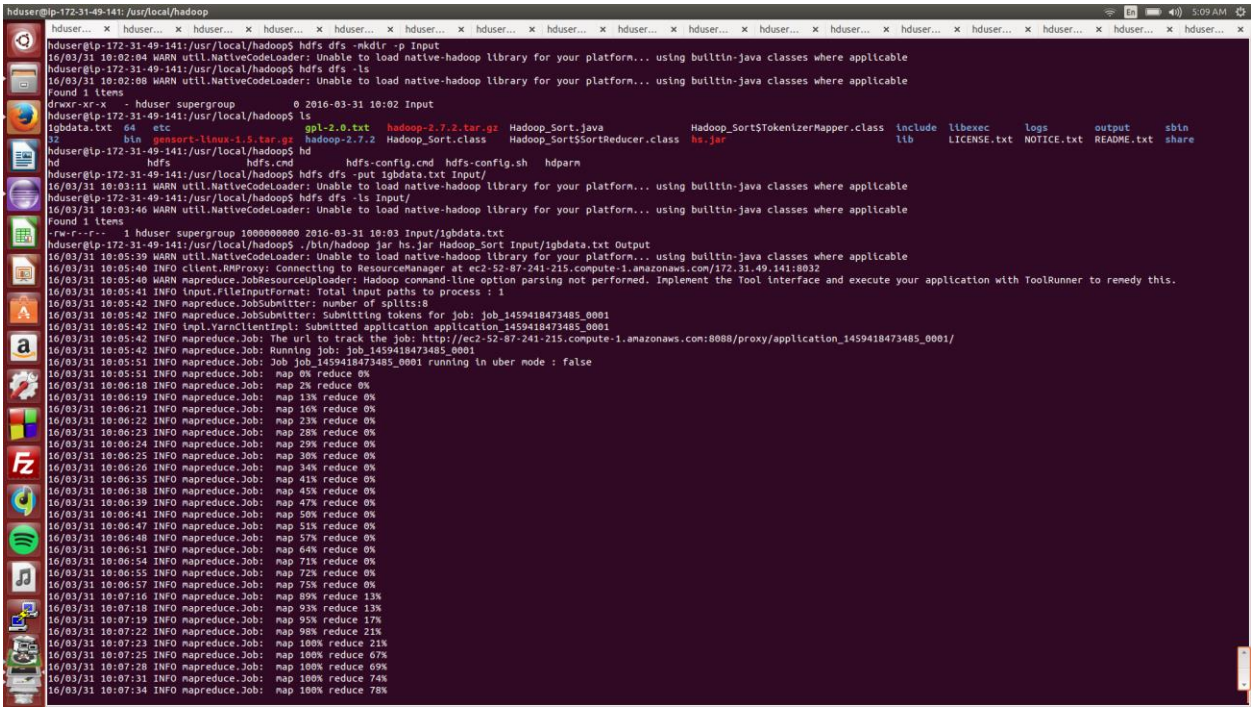


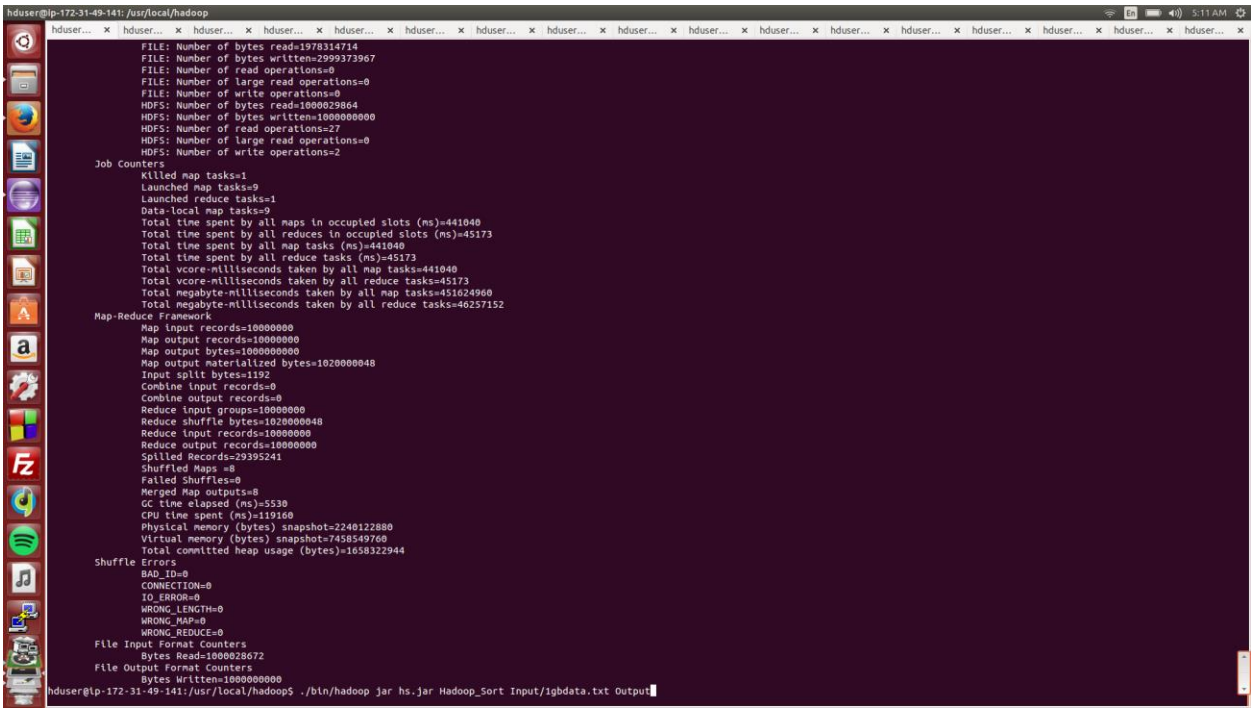
CS553 Programming Assignment #2
Sort on Hadoop/Spark



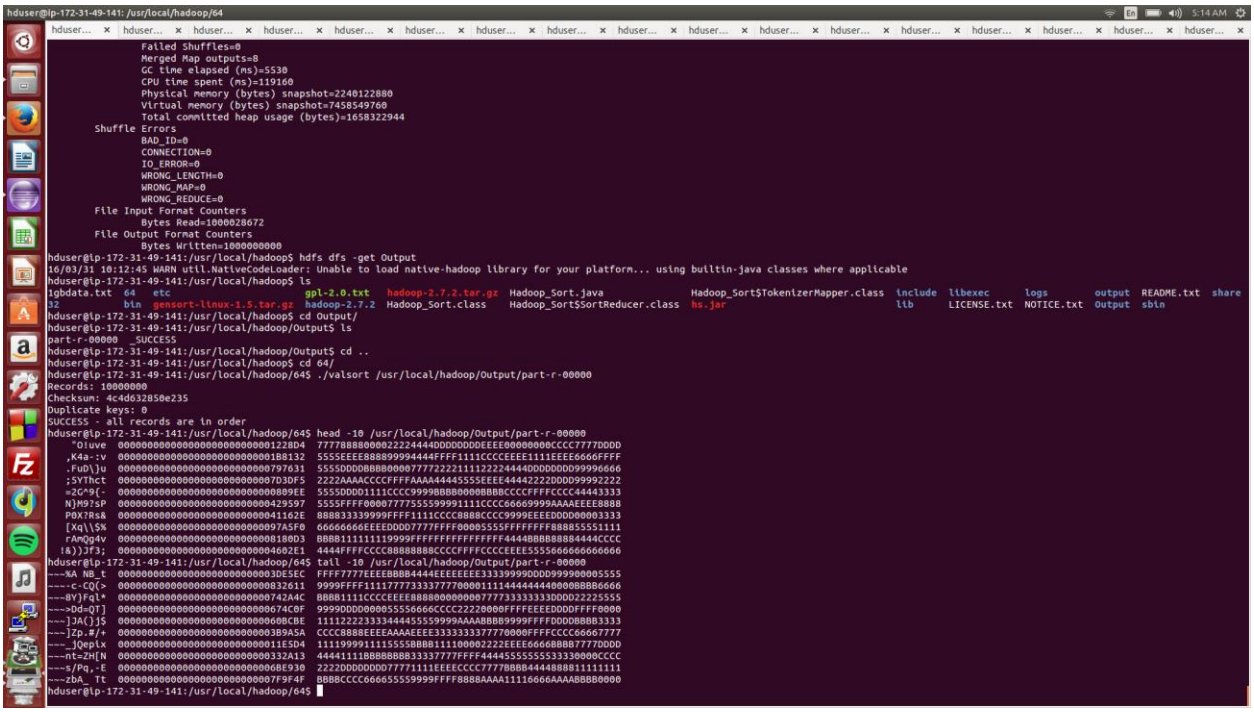


CS553 Programming Assignment #2
Sort on Hadoop/Spark



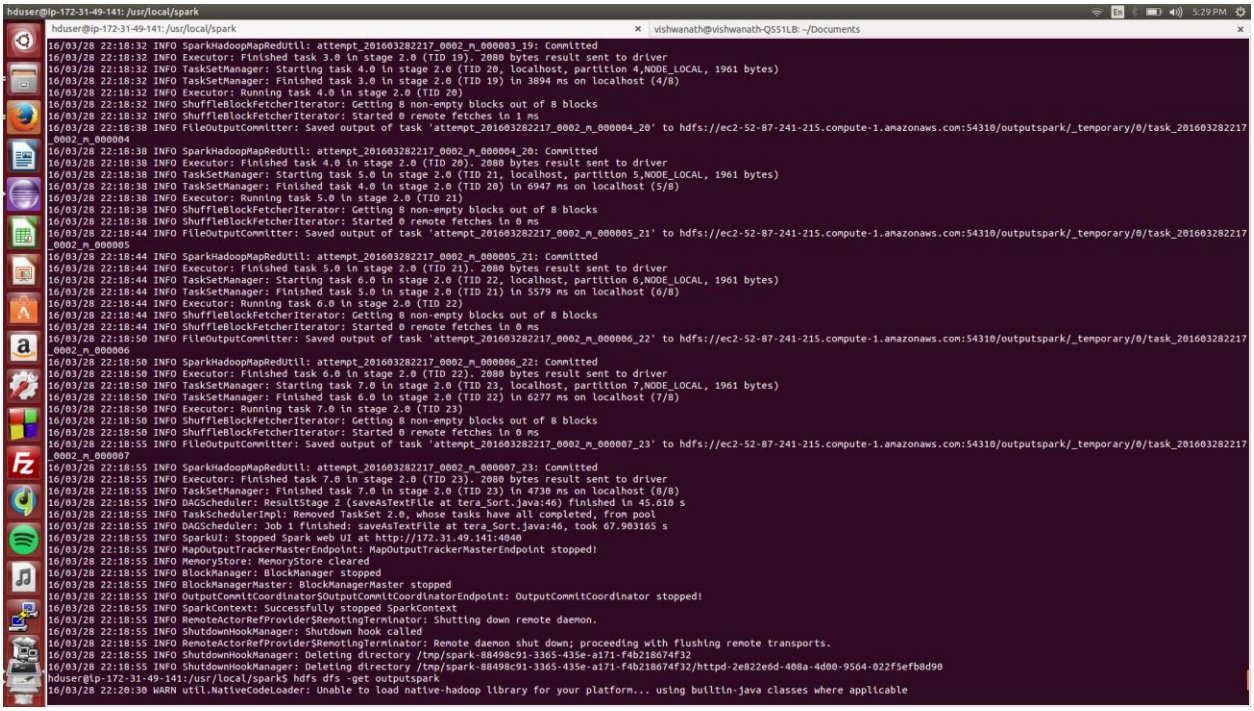


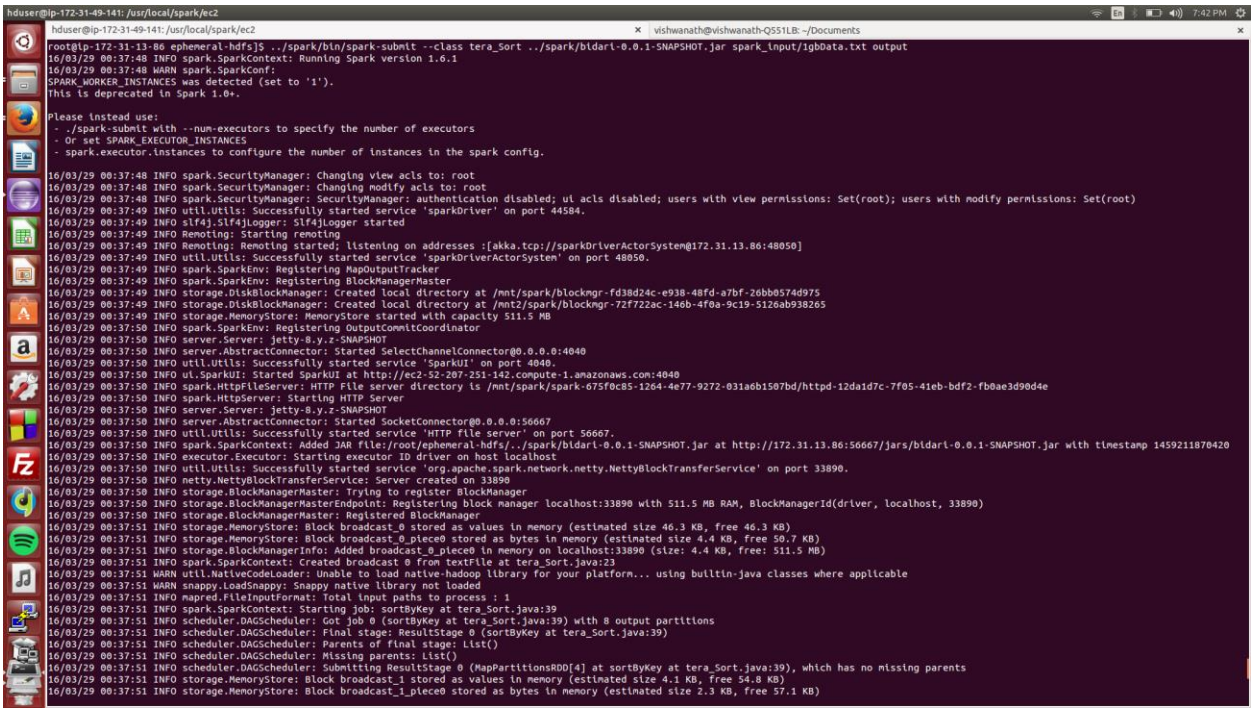
CS553 Programming Assignment #2
Sort on Hadoop/Spark



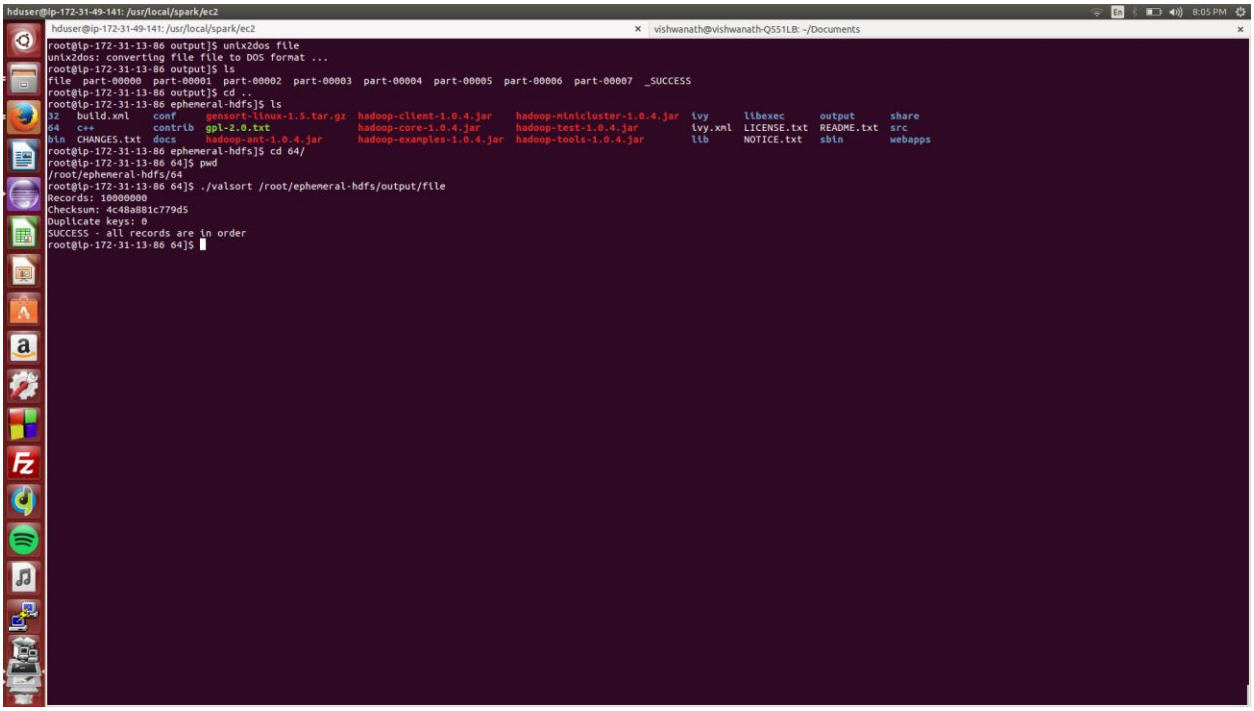
3. Spark

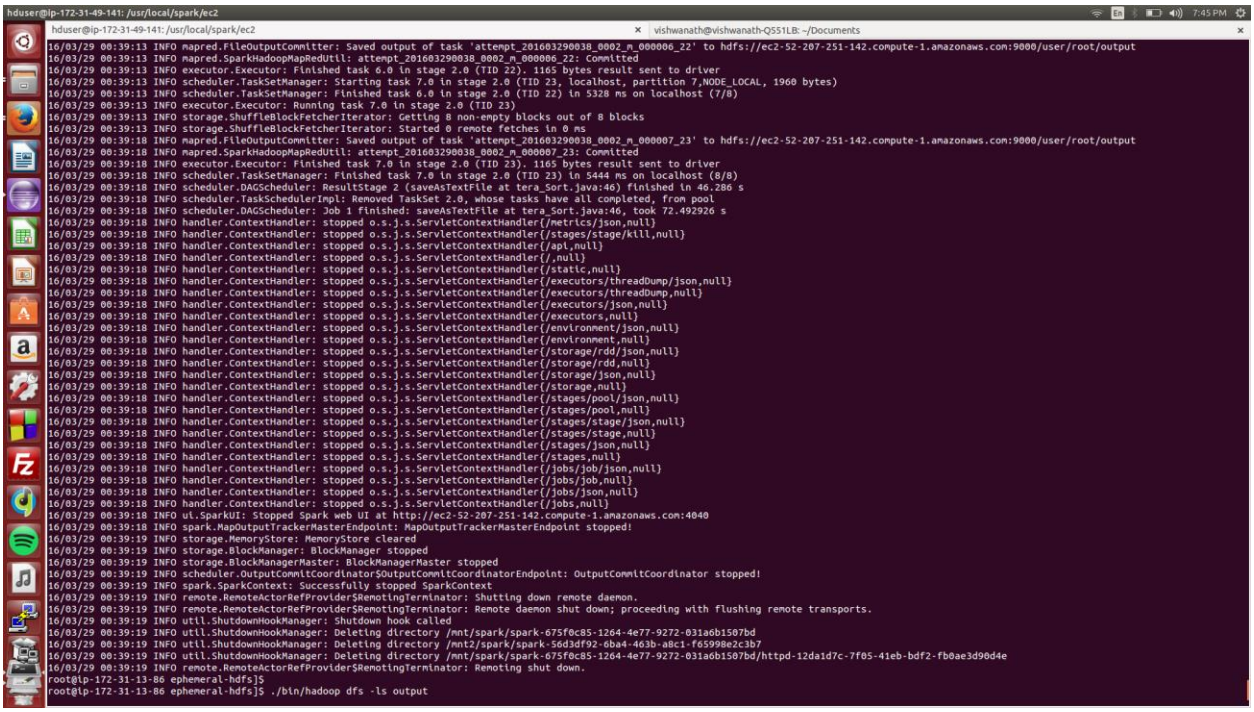
CS553 Programming Assignment #2
Sort on Hadoop/Spark



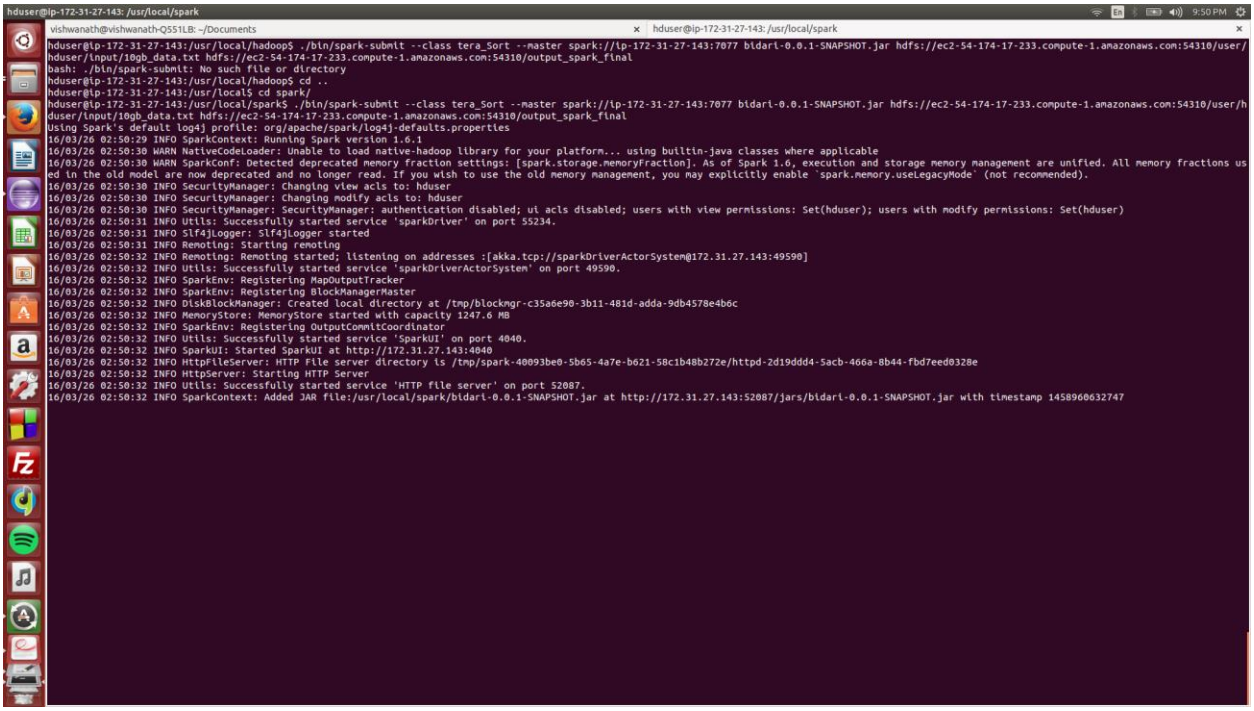


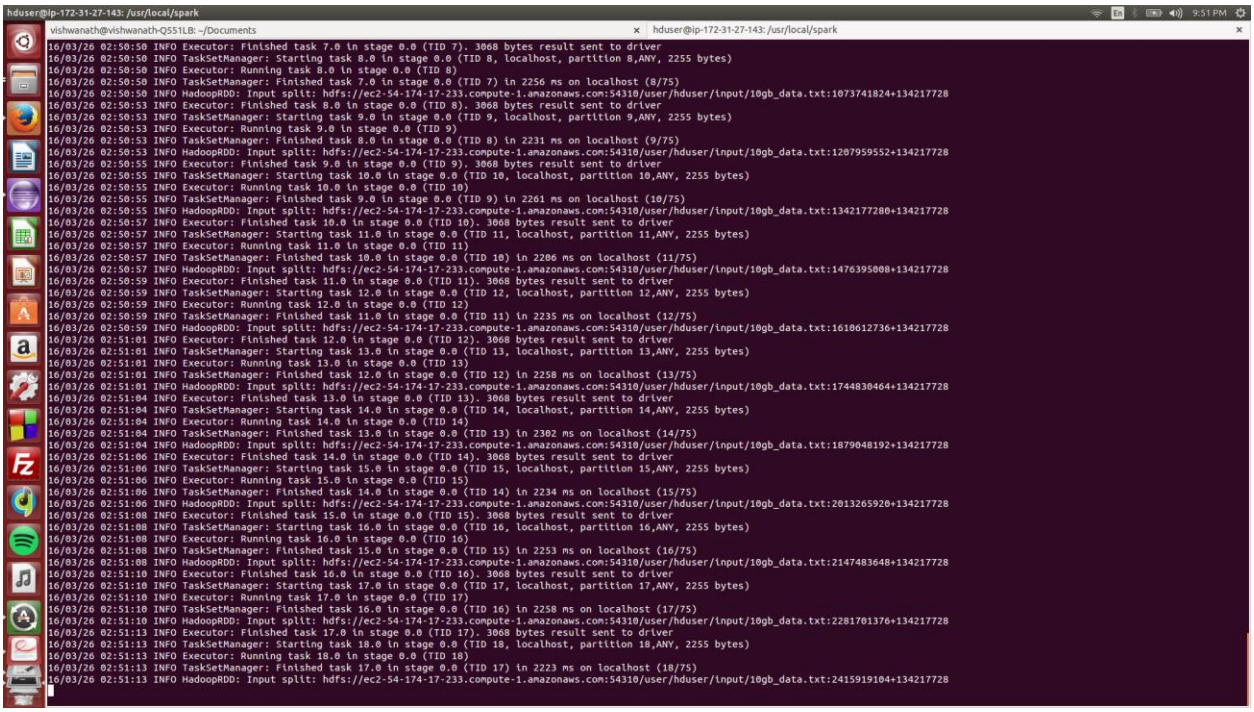
CS553 Programming Assignment #2
Sort on Hadoop/Spark



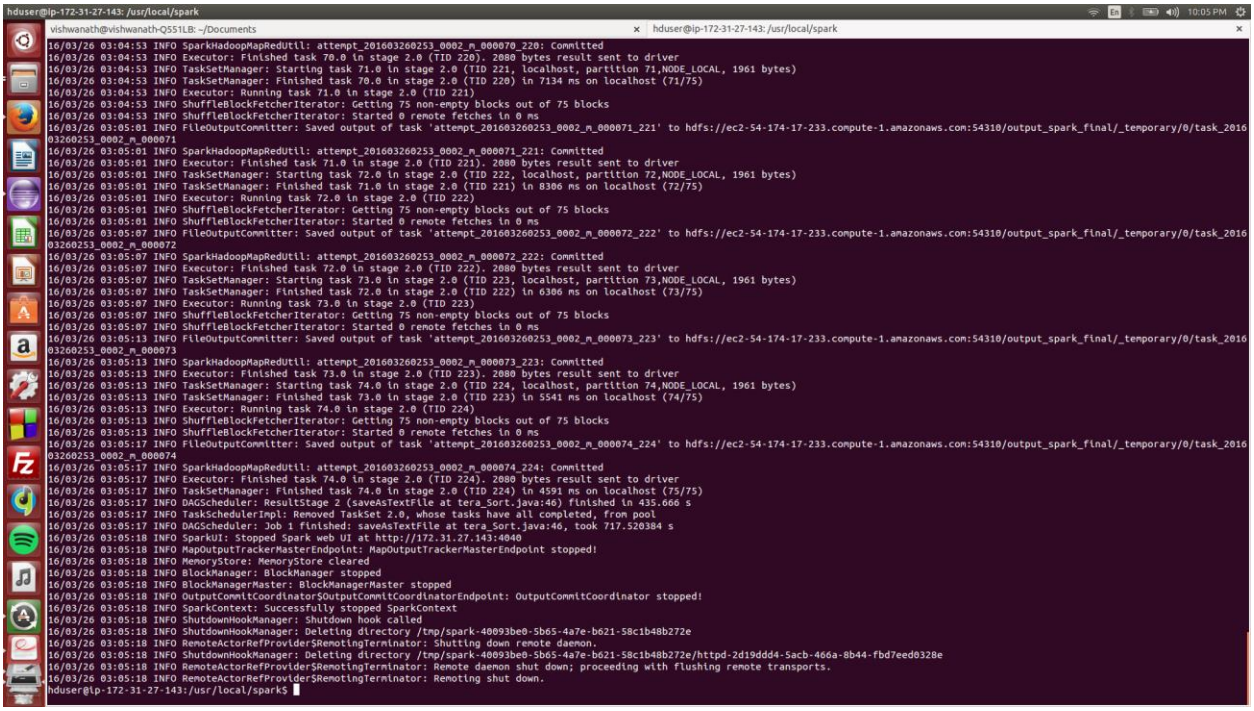


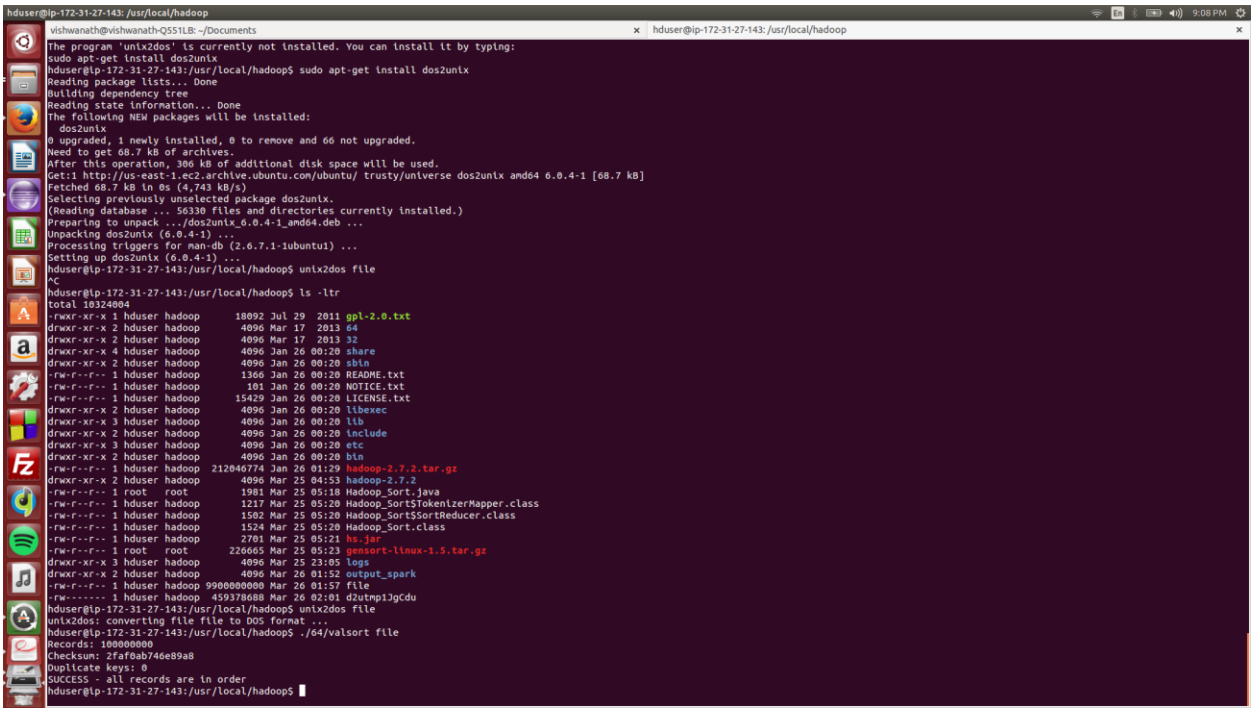
CS553 Programming Assignment #2
Sort on Hadoop/Spark



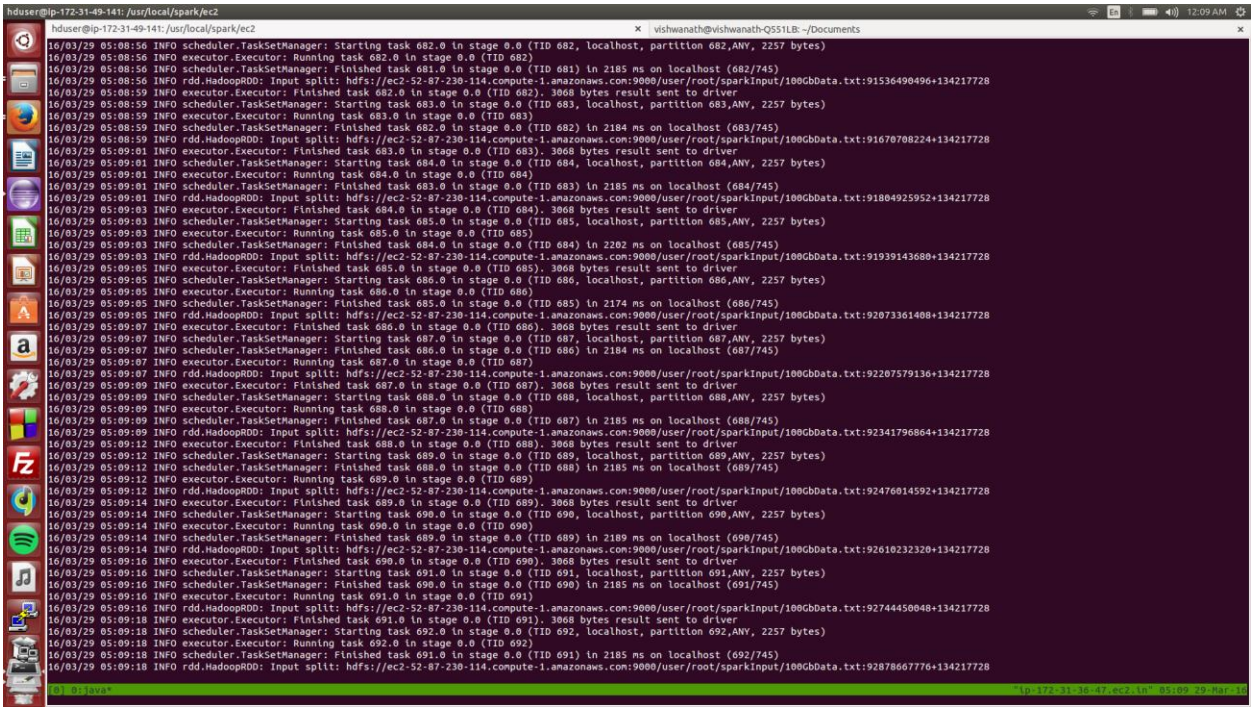


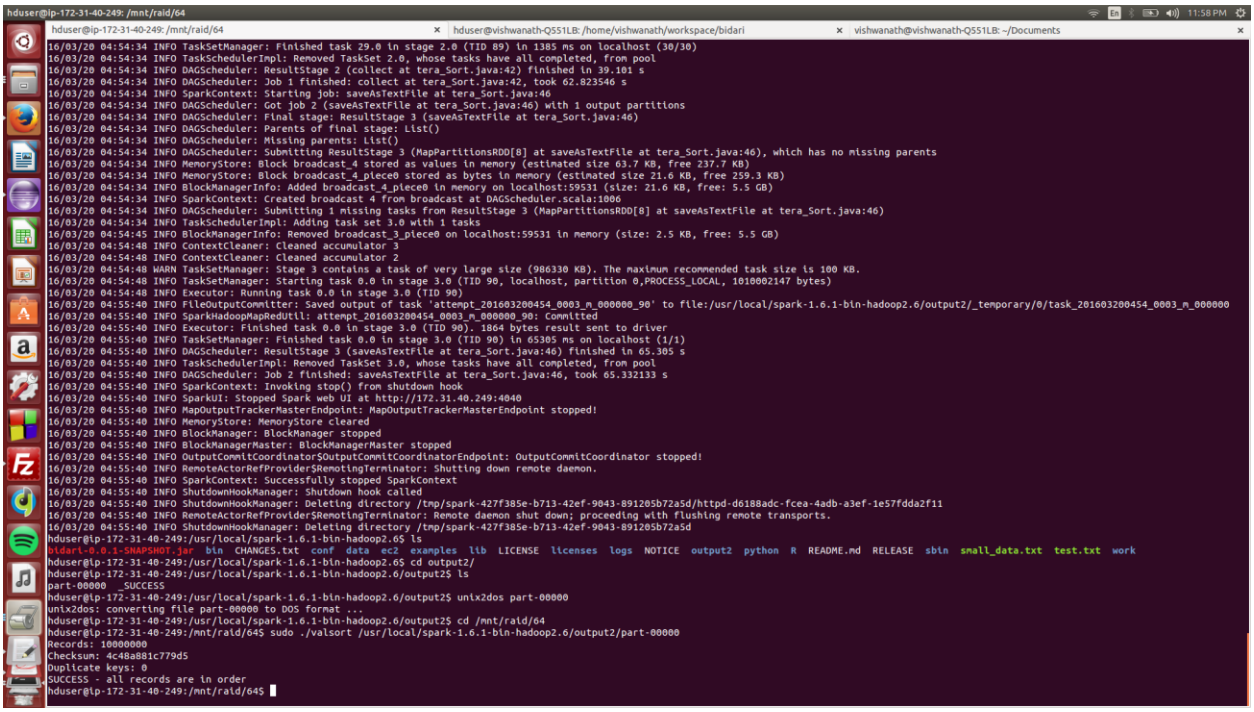
CS553 Programming Assignment #2
Sort on Hadoop/Spark



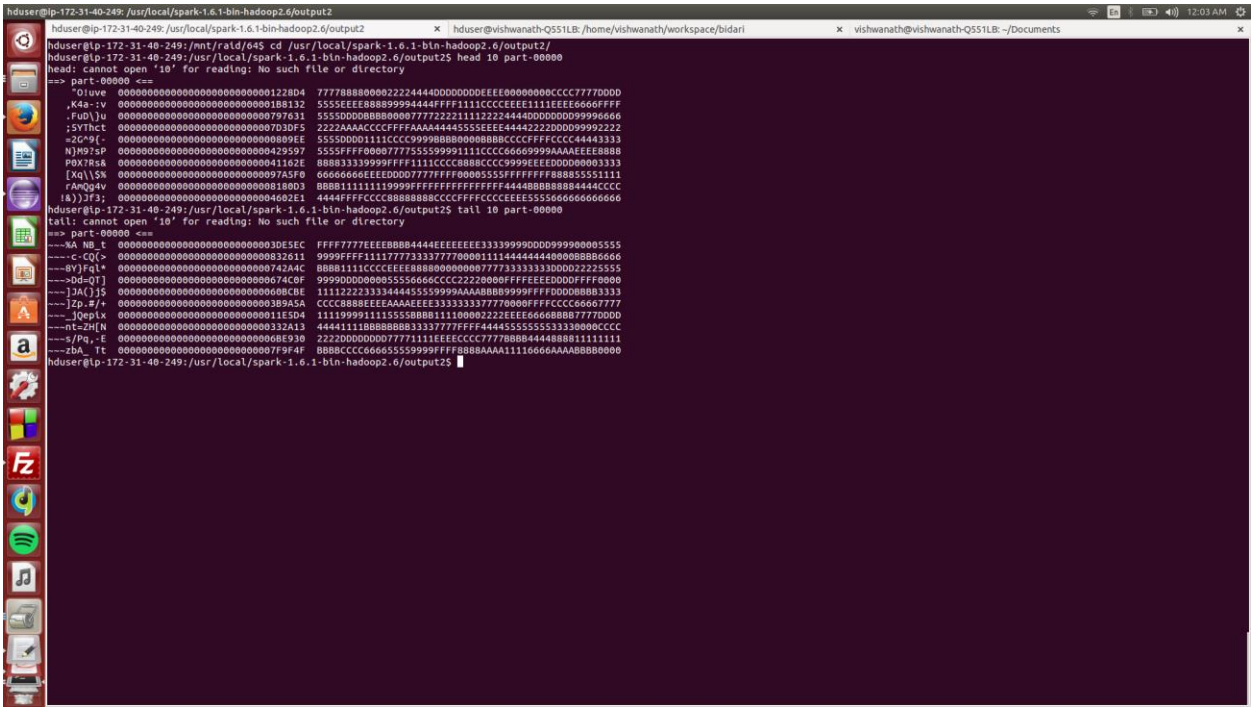


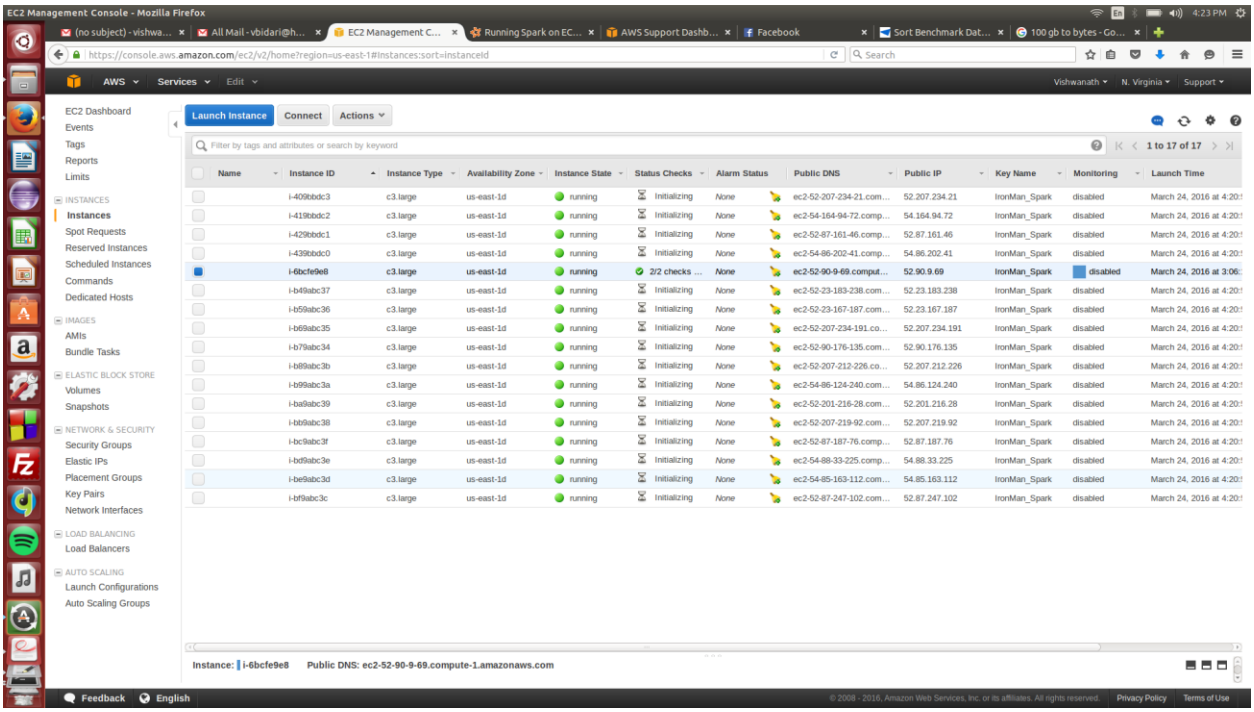
CS553 Programming Assignment #2
Sort on Hadoop/Spark



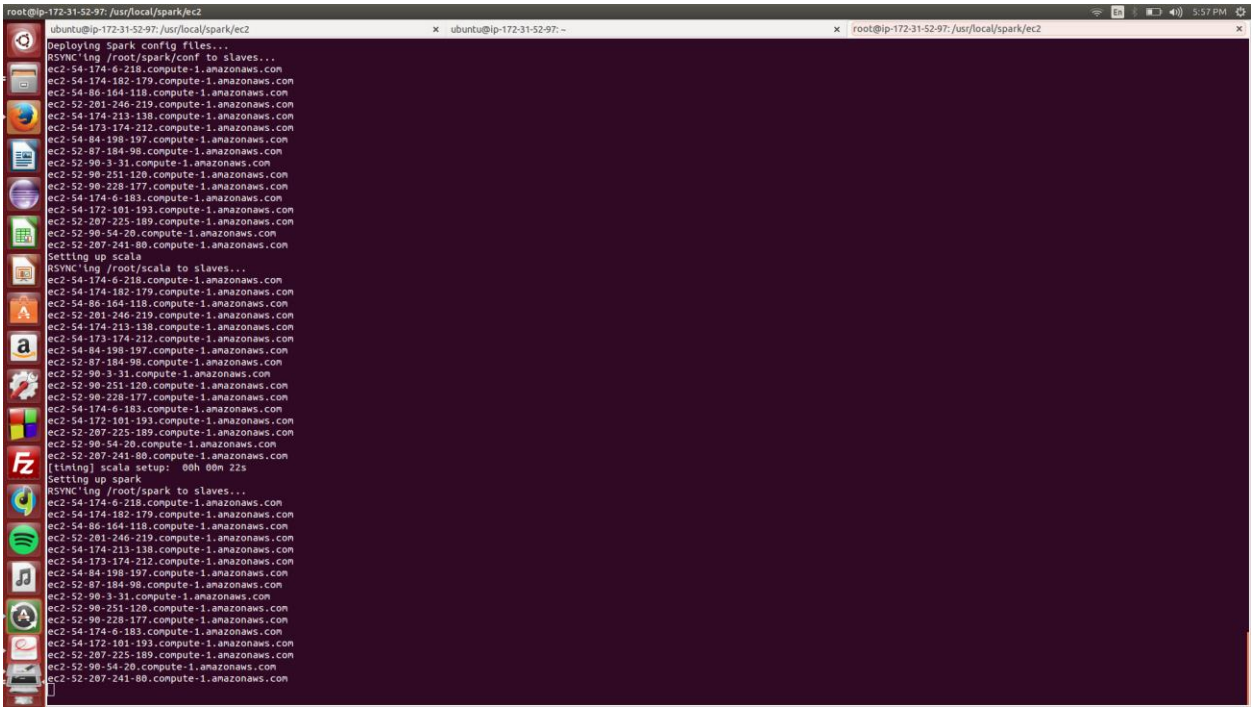


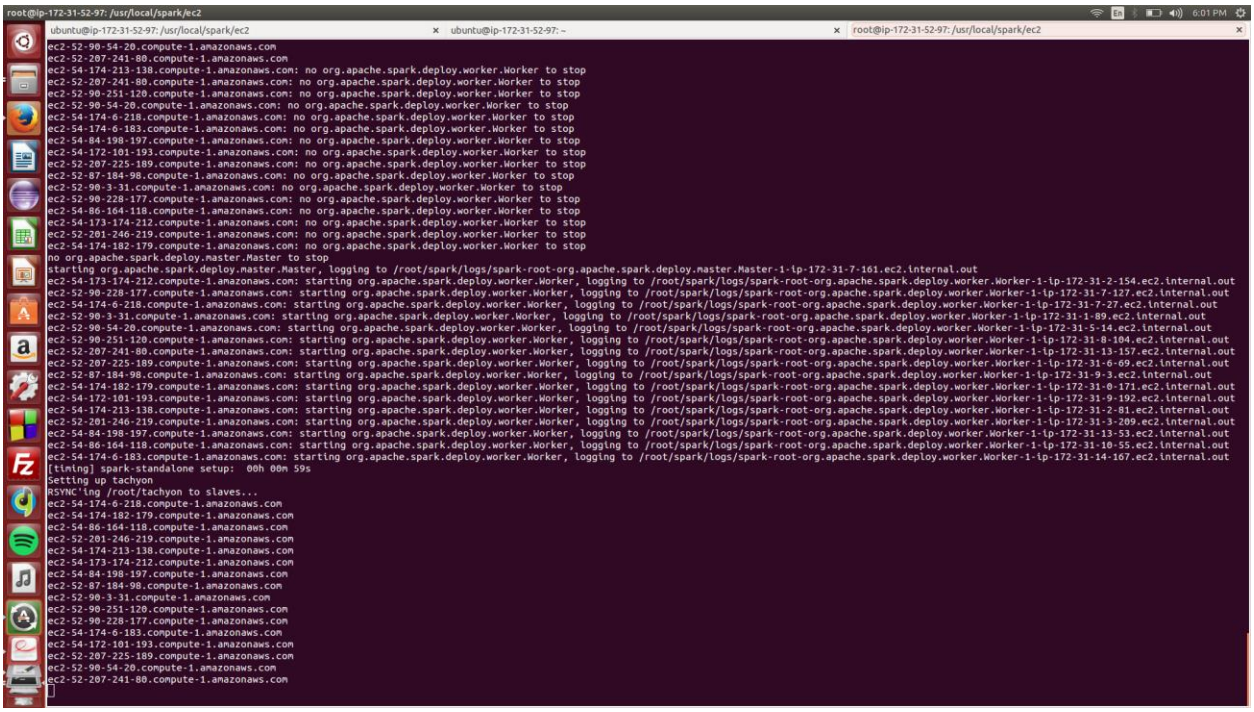
CS553 Programming Assignment #2
Sort on Hadoop/Spark



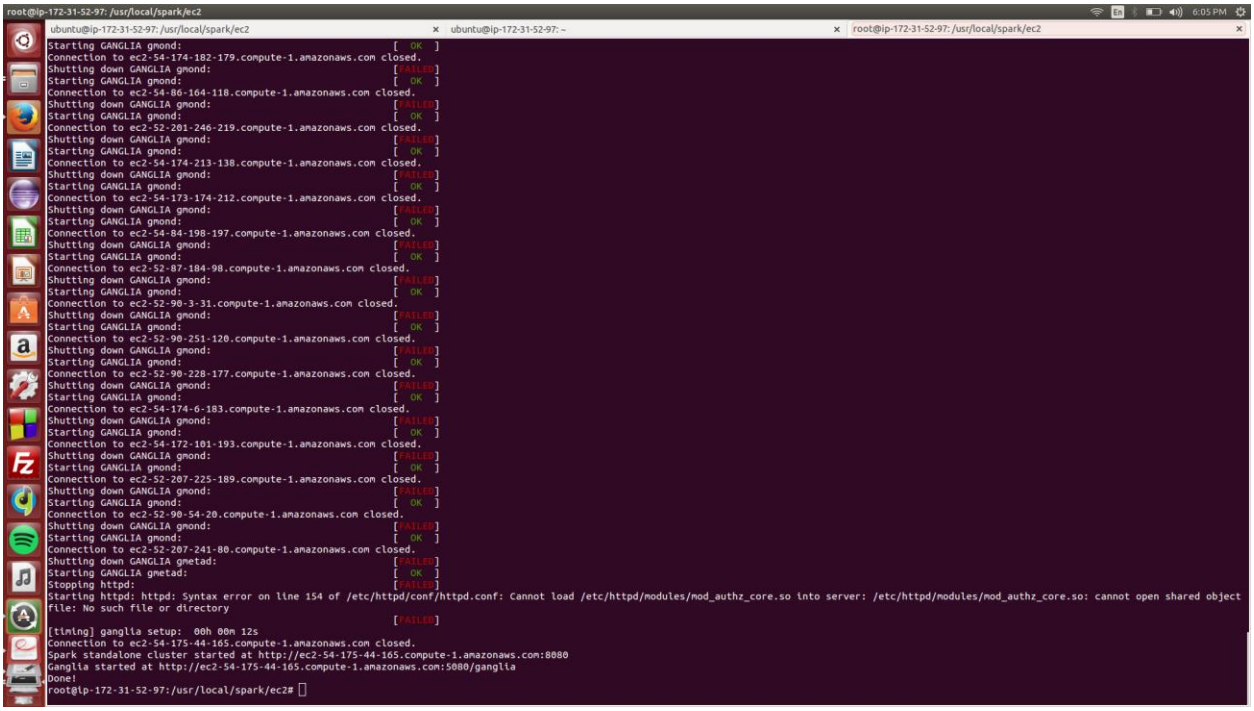


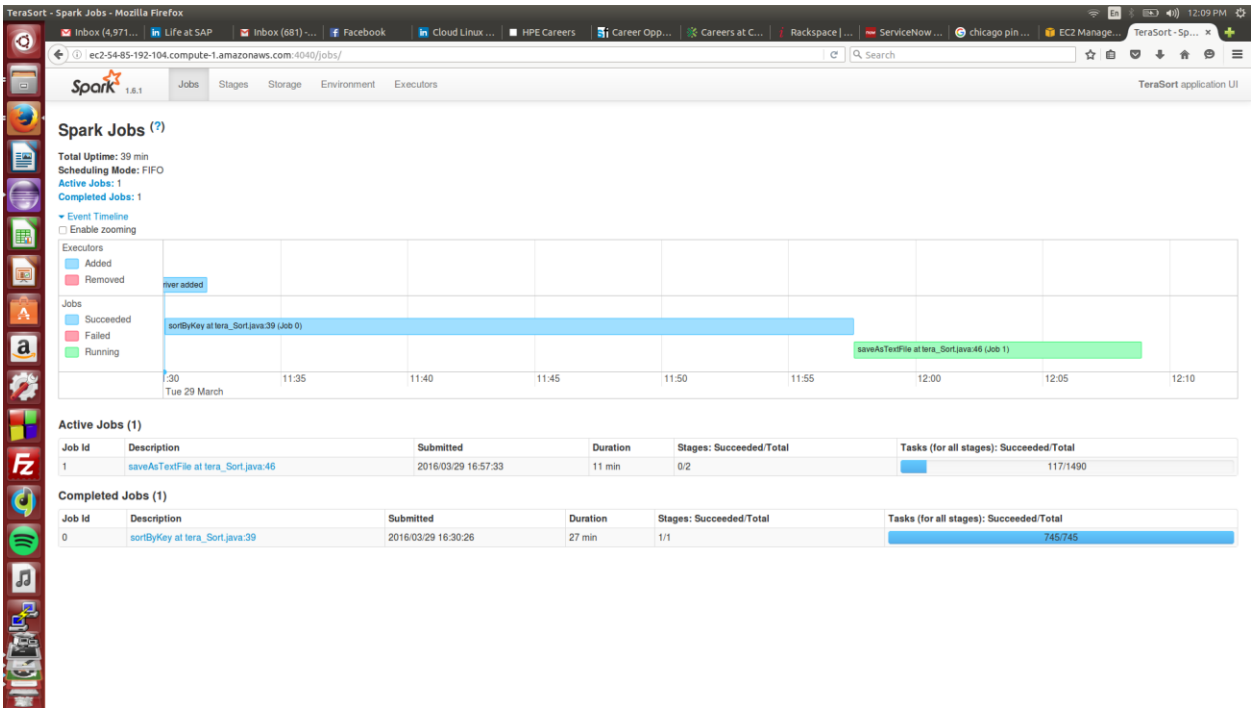
CS553 Programming Assignment #2
Sort on Hadoop/Spark





CS553 Programming Assignment #2
Sort on Hadoop/Spark





CS553 Programming Assignment #2
Sort on Hadoop/Spark