CS 553
CLOUD COMPUTING
PROGRAMMING ASSIGNMENT-2

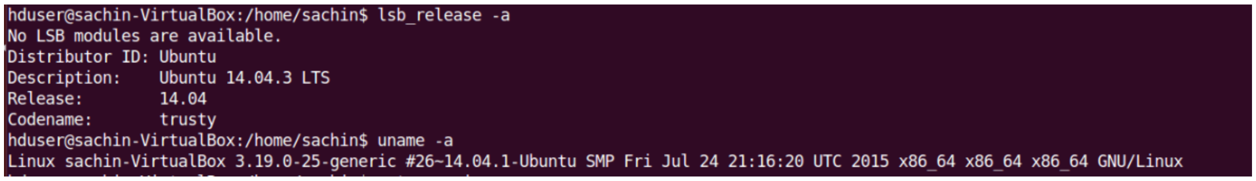
SUBMITTED BY :

SACHIN KRISHNA MURTHY

CWID : A20354077

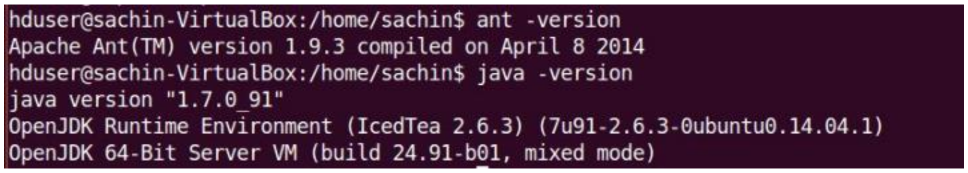
DESIGN DOCUMENT

Ubuntu Version :



```
hduser@sachin-VirtualBox:/home/sachin$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.3 LTS
Release:        14.04
Codename:       trusty
hduser@sachin-VirtualBox:/home/sachin$ uname -a
Linux sachin-VirtualBox 3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

Ant and Java Version :



```
hduser@sachin-VirtualBox:/home/sachin$ ant -version
Apache Ant(TM) version 1.9.3 compiled on April 8 2014
hduser@sachin-VirtualBox:/home/sachin$ java -version
java version "1.7.0_91"
OpenJDK Runtime Environment (IcedTea 2.6.3) (7u91-2.6.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.91-b01, mixed mode)
```

SHARED MEMORY

The Shared Memory program has been implemented in Java.

➢ **Working Procedure:**

- Read the input and output file names.
- Read the input file into a List of List.
- For each List, inside the bigger list
  - Sort the list using mergesort with thread
- Merge internal lists into a new List of List
  - use finalmerge method
- Read list by list into output file
- Calculation of throughput value using the formula :

  Throughput (MB/sec) = Dataset Size (MB) / Sort Time (sec)

  Merge sort

- Divide the array in two sub arrays - left and right
- Make a recursive call for the left array

- If array size = 1; return
- Else go to step1
- Make a recursive call for the right array
- If array size = 1; return
- Else go to step1
- Merge the arrays.

FinalMerge

- Set index of all list to 0
- For each list, take indexed element and find least of them
- Put least element into new list
- Increment index of least element's list
- If new list size limit exceeds
- Insert new list into list of list
- Create another new list
- Goto step 1
- If all lists are empty
- Insert new list into list of list
- Return list of list

HADOOP

Version Used : Hadoop 2.7.2

Objective : To sort 10gb input file on 1 node and 100gb input file on 16 nodes with Hadoop installed on them, calculation of time taken and throughput for the sort program and comparison of the results of 1 node to 16 nodes.

Configuration of Hadoop on a Single Node :

- The first step is to install java.

```
sachin@sachin-VirtualBox:~$ cd ~
```

```
# Update the source list
sachin@sachin-VirtualBox:~$ sudo apt-get update

sachin@sachin-VirtualBox:~$ sudo apt-get install default-jdk

sachin@sachin-VirtualBox:~$ java -version
java version "1.7.0_65"
```

- The next step is to add a dedicated Hadoop user.

```
sachin@sachin-VirtualBox:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1002) ...
Done.

sachin@sachin-VirtualBox:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
            Other []:
Is the information correct? [Y/n] Y
```

- Then we have to install the SSH.

  SSH has two components :

  - **ssh** : The command we use to connect to remote machines - the client.
  - **sshd** : The daemon that is running on the server and allows clients to connect to the server.

  The following command can be used to install the SSH.

```
sachin@sachin-VirtualBox:~$ sudo apt-get install ssh
sachin@sachin-VirtualBox:~$ which ssh
/usr/bin/ssh
sachin@sachin-VirtualBox:~$ which sshd
/usr/sbin/sshd
```

- The next step is to create and setup SSH certificates.
  You are required to enter the password for the hduser that you created and then we need to generate a rsa key pair as shown in the below commands.

```
sachin@sachin-VirtualBox:~$ su hduser
Password:
sachin@sachin-VirtualBox:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop
```

```
The key's randomart image is:
+--[ RSA 2048]----+
|       .oo.o    |
|      . .o=. o  |
|     . + .  o . |
|      o =    E  |
|       S +      |
|        . +     |
|         O +    |
|          O o   |
|           o..  |
+-----------------+
hduser@sachin-VirtualBox:/home/k$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- Checking if SSH works

```
hduser@ sachin-VirtualBox:/home/sachin$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)
...
```

- Installing Hadoop

```
hduser@sachin-VirtualBox:~$wget   http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
hduser@laptop:~$ tar xvzf hadoop-2.7.2.tar.gz
```

- Here we are moving the installed Hadoop to the /usr/local/hadoop/ directory using the following command:

```
hduser@laptop:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
```

- In the next step we have to set up the following configuration files:

  - ~/.bashrc
  - /usr/local/hadoop/etc/hadoop/hadoop-env.sh
  - /usr/local/hadoop/etc/hadoop/core-site.xml
  - /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
  - /usr/local/hadoop/etc/hadoop/hdfs-site.xml

Initially we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:

```
hduser@sachin-VirtualBox: update-alternatives --config java

There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

Nothing to configure.
```

Next, we can add the following content at the end of ~/.bashrc file.

```
hduser@sachin-VirtualBox:~$ vim ~/.bashrc


#HADOOP VARIABLES START

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP_INSTALL=/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"
#HADOOP VARIABLES END

hduser@ sachin-VirtualBox:~$ source ~/.bashrc
```

Also that JAVA_HOME should be set as the path just before the '.../bin/':

```
hduser@ sachin-VirtualBox:~$ javac -version
javac 1.7.0_75

hduser@ sachin-VirtualBox:~$ which javac
/usr/bin/javac

hduser@ sachin-VirtualBox:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-7-openjdk-amd64/bin/javac
```

- We need to set JAVA_HOME by modifying hadoop-env.sh file.

```
hduser@sachin-VirtualBox:~$vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

- **Core-site.xml file**

```
hduser@ sachin-VirtualBox:~$ sudo mkdir -p /app/hadoop/tmp
hduser@ sachin-VirtualBox:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

We need to add the following contents in between the <configuration> </configuration> tag.

```
hduser@sachin-VirtualBox:~$vim /usr/local/hadoop/etc/hadoop/core-site.xml


<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://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

- **mapred-site.xml file**

```
hduser@sachin-VirtualBox:~$    cp    /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

We need to add the following contents in between the <configuration> </configuration> tag.

```
<configuration>
 <property>
```

```
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

- **hdfs-site.xml file :**

  For this we need to create two directories which will contain namenode and datanode for this Hadoop installation.

```
hduser@sachin-VirtualBox:~$sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
hduser@sachin-VirtualBox:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
hduser@sachin-VirtualBox:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

We need to add the following contents in between the <configuration> </configuration> tag.

```
hduser@sachin-VirtualBox:~$vim/usr/local/hadoop/etc/hadoop/hdfs-site.xml



<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <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>
```

- After setting up the configuration files, we need to format the Hadoop file system by using the below command :

```
hduser@ sachin-VirtualBox:~$ hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = laptop/192.168.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop
...
STARTUP_MSG:   java = 1.7.0_65
************************************************************/
```

```
15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]

15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]

15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f

15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.

15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-check=true

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will start around 2015 Apr 18 14:43:10

15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap

15/04/18 14:43:10 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB

15/04/18 14:43:10 INFO util.GSet: capacity      = 2^21 = 2097152 entries

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication         = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication             = 512

15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication             = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplicationStreams      = 2

15/04/18 14:43:10 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks  = false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: encryptDataTransfer        = false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxNumBlocksToLog          = 1000

15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner             = hduser (auth:SIMPLE)

15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup          = supergroup

15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled = true
```

```
15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false

15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map INodeMap

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB

15/04/18 14:43:11 INFO util.GSet: capacity      = 2^20 = 1048576 entries

15/04/18 14:43:11 INFO namenode.NameNode: Caching file names occuring more than 10 times

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map cachedBlocks

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB

15/04/18 14:43:11 INFO util.GSet: capacity      = 2^18 = 262144 entries

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 
0.9990000128746033

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 
0

15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 
30000

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and 
retry cache entry expiry time is 600000 millis

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB

15/04/18 14:43:11 INFO util.GSet: capacity      = 2^15 = 32768 entries

15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false

15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true

15/04/18 14:43:11 INFO namenode.NNConf: Maximum size of an xattr: 16384

15/04/18 14:43:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-130729900-
192.168.1.1-1429393391595

15/04/18 14:43:12 INFO common.Storage: Storage directory 
/usr/local/hadoop_store/hdfs/namenode has been successfully formatted.
```

```
15/04/18 14:43:12 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0

15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0

15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:

/************************************************************

SHUTDOWN_MSG: Shutting down NameNode at laptop/192.168.1.1

************************************************************/
```

- **<u>Hadoop Configuration on 16 nodes :</u>**

After configuring Hadoop on a single node, we need to create a image of the same on 16 instances. So out of 17 instances we need to use one as a master to run the program and the rest 16 as slaves.

Here you will be assigning the IP address of the master in the core-site.xml, mapred-site.xml and yarn file of the 16 slaves. Also we need to assign the respective slave IP address of each slave in their slave file. In the master node, we need to specify the master IP and all the 16 slaves IP address in the slave file of the master.

<u>Problem Faced While configuring :</u>

I was trying to sort 100gb on 16 nodes and I had done all the necessary configurations. Also when I run start-dfs.sh on master node I was able to see that it starts all the namenodes and 16 datanodes respectively. Thereafter when I am putting the file into hdfs file system, I was getting an error saying “hdfs could only be replicated to 0 nodes instead of 1”.

➢ **<u>Working Procedure:</u>**

<u>Main class</u>

- Set configurations
- Get job Instance
- Set mapper class
- Set combiner class
- Set reducer class
- Set mainclass jar

- Calculate and display throughput.

<u>Mapper</u>

- Read input file part by part into text.
- Sort text of parts.
- Write sorted text.

<u>Reducer</u>

- Read sorted texts.
- Merge texts into single text.
- Write merged text into output file.

➢ **Master Node :**

The Master node consists of namenode and jobtracker where :

- The Master (NameNode) manages the file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes along with regulating access to files by clients.
- Master (Jobtracker) is the point of interaction between users and the map/reduce framework. When a map/reduce job is submitted, Jobtracker puts it in a queue of pending jobs and executes them on a first-come/first-served basis and then manages the assignment of map and reduce tasks to the tasktrackers.

➢ **Slave Node :**

The Slave node consists of datanode and Tasktraker where:

- Slaves (DataNodes) are responsible for performing read and write requests from the file system's clients along with performing block creation, deletion, and replication upon instruction from the Master (NameNode).
- Slaves (tasktracker) executes tasks upon instruction from the Master (Jobtracker) and also handle data motion between the map and reduce phases.

➢ We need to set unique available ports because to reduce traffic on single port, since each node will be doing its own scheduled task. If we use the same port number we will encounter connection bind exception.

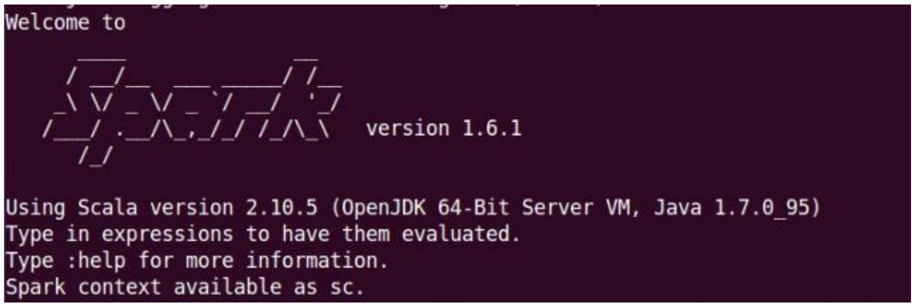
- The number of Mappers and Reducers can be set by configuring JobConf variables in the code:
  job.setNumMapTasks(num);
  job.setNumReduceTasks(num);

  where num is any value for the number of mappers and reducers.

- Also we can infer that Hadoop performance is better in case of single node and Spark performance is better in case of multiple nodes when compared with shared memory and Hadoop.

SPARK

Version :



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Scala version 2.10.5 (OpenJDK 64-Bit Server VM, Java 1.7.0_95)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc.
```

Objective : To sort 10gb input file on 1 node and 100gb input file on 16 nodes with Spark installed calculation of time taken and throughput for the sort program and comparison of the results of 1 node to 16 nodes.

Configuring Spark on a single node :

- We need to install Java before we install Spark.
- The next step is to download and install scala. The following command can be used for extracting the scala tar file.
  $tar xvf scala-2.11.6.tgz

- We need to move the scala software to the respective directory.
  mv scala-2.11.6 /usr/local/scala

- Set path for scala:

```
$export PATH=$PATH:/usr/local/scala/bin
```

- The next step is to download and install apache spark. The following command can be used for extracting the spark tar file:
  ```
  $tar xvf spark-1.3.1-bin-hadoop2.6.tgz
  ```

- Then we need to move the spark software files to the respective directory.
  ```
  mv spark-1.3.1-bin-hadoop2.6 /usr/local/spark
  ```

- Finally we need to set up the environment for spark by adding the following line to ~/.bashrc file.
  ```
  export PATH=$PATH:/usr/local/spark/bin
  ```

- We can use the following command for sourcing the ~/.bashrc file.

  ```
  $source ~/.bashrc
  ```

Configuring Spark on 16 nodes:

- Initially we need to get the Access Key Id and Secret Access Key Id from the Amazon.
- We need to set the environment variables for the AWS access key and secret access key that we are going to use.
  ```
  export AWS_ACCESS_KEY_ID=<Access Key Here>
  export AWS_SECRET_ACCESS_KEY=<Secret Access Key Here>
  ```

- In the downloaded spark folder we need to navigate to the folder named ec2 and run the following command:
  ```
  ./spark-ec2 --key-pair=YOUR_KEY_NAME --identity-file=YOUR_KEY.pem --region=us-west-1 --slaves=16 --instance-type=c3.large --ebs-vol-size=1000 --spot-price=0.03 launch spark
  ```

- By executing the above command 16 slave instances along with a master instance will be launched.
- You can go ahead in executing the program once the connection is established.

Working Procedure :

- Read the input file.
- Map string into key value format.
- First 10 chars form key, while rest form value.
- Sort the map
- Write sorted map into output file
- Calculate and display throughput.