SPARK CLUSTER INSTALLATION

Software Requirements for this setup-
Ubuntu 12.04
Apache Spark 1.0.0
Apache Hadoop 2.2.0
Python 3.4

Setting up your Spark Cluster-

First install Spark as a single node cluster on all systems.

Prerequisites:
Install Java OpenJDK.
sudo apt-get install openjdk-7-jdk

The jdk will be downloaded in the following folder for Ubuntu Linux 64 bit.
Finding the location of Java installation on 64 bit Ubuntu Linux
/usr/lib/jvm/java-7-openjdk-amd64
$ pwd
/usr/lib/jvm/java-7-openjdk-amd64

Checking if Java has been installed correctly
$ java -version
java version "1.7.0_51"
OpenJDK Runtime Environment (IcedTea 2.4.4) (7u51-2.4.4-0ubuntu0.12.04.2)
OpenJDK 64-Bit Server VM (build 24.45-b08, mixed mode)

Add new group hadoop
$ sudo addgroup spark
Adduser sparkuser in group spark
$ sudo adduser --ingroup spark sparkuser

User privilege specification
root ALL=(ALL:ALL) ALL
sparkuser ALL=(ALL:ALL) ALL

Configuring SSH
Generate SSH key
$ ssh-keygen -t rsa -P ""
Add newly generated key to authorized keys
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

Testing SSH connect
$ ssh localhost

Installing Spark single node:

Download the latest version of Scala programming language and extract it
$wget http://www.scala-lang.org/files/archive/scala-2.9.3.tgz
$tar xzvf scala-2.9.3.tgz

Download the latest version of Spark and install it(in this case Spark 1.0.0)
$ git clone https://github.com/apache/spark.git

Change permissions of the folder
$ sudo chown -R sparkuser:spark spark

Installing SBT.We'll use Scala SBT (Simple Build Tool) for building our standalone app.
$ wget http://dl.bintray.com/sbt/native-packages/sbt/0.13.2/sbt-0.13.2.tgz

$ tar zxvf sbt-0.13.2.tgz

Change ownership of sbt folder. $ sudo chown -R sparkuser:spark sbt

Add the following line in .bashrc : export PATH=$PATH:/home/sparkuser/sbt/bin

Building spark wit Hadoop 2.2.0 version.Run the folowing command in the spark folder.
$ SPARK_HADOOP_VERSION=2.2.0 SPARK_YARN=true sbt/sbt clean assembly

Check the working of the installation :

Test the working by starting the spark-shell and running a simple word count program.
$ cd spark/bin
$ ./spark-shell

scala> val textFile=sc.textFile("README.md")

textFile: org.apache.spark.rdd.RDD[String] = MappedRDD[1] at textFile at <console>:12

scala> textFile.count

res0: Long = 126

scala> textFile.filter(_.contains("the")).count

res1: Long = 28

scala> exit

Starting the master and worker daemons :

$ cd spark/sbin

$ ./start-all.sh

Use jps command to check that the master and worker daemons are running.
Go to the home folder and execute the following command.
$vim .bashrc
Add these following lines anywhere in the file
export SCALA_HOME=/home/sparkuser/scala
export PATH=$PATH:$SCALA_HOME/bin
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export SPARK_HOME=/home/sparkuser/spark
export PATH=$PATH:$SPARK_HOME/bin
export PATH=$PATH:/home/sparkuser/sbt/bin
export HADOOP_PREFIX=/home/sparkuser/hadoop

export
SPARK_JAR=/home/sparkuser/spark/assembly/target/spark-assembly-1.1.0-SNAPSHOT-hado
op2.2.0.jar/scala-2.10

$source .bashrc

Setting up Standalone Spark Cluster

We begin with updating the hosts file
$ sudo vim /etc/hosts

Update the hosts file with the IP addresses of the worker machines along with the master machine IP on all the machines in the cluster.
For example:
192.168.43.45 master
192.168.43.46 slave1
192.168.43.47 slave2
192.168.43.48 slave3
SSH access
To allow the master to access the slaves and itself through passwordless ssh.
Run the following commands on master only.
$ssh-copy-id -i $HOME/.ssh/id_rsa.pub sparkuser@worker

$ssh-copy-id -i $HOME/.ssh/id_rsa.pub sparkuser@master

Next, we move onto changing configurations in Spark.

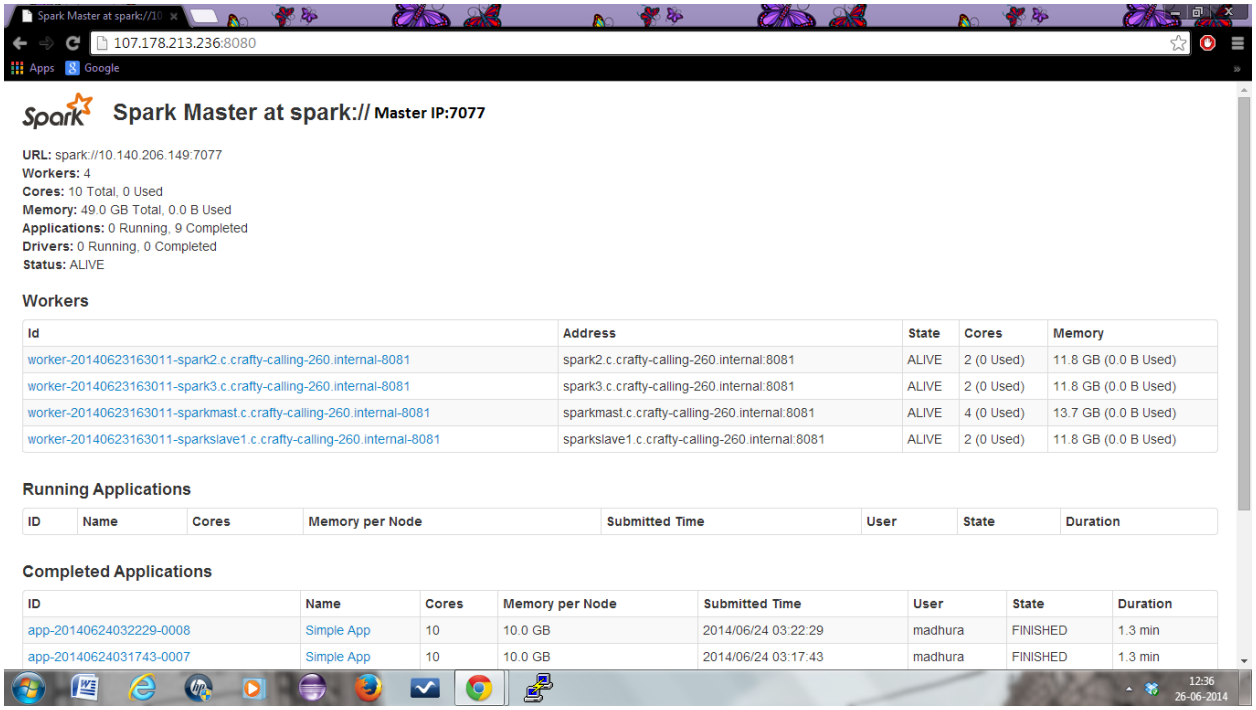
Do this for all machines in the cluster.
Open spark-env.sh
$ vim spark/conf/spark-env.sh
Add the following lines anywhere in the file
export SPARK_MASTER_IP=<Master IP>
export SPARK_MASTER_PORT=7077
export SPARK_WORKER_PORT=8081
Also set appropriate values for number of executors and their memory.

Open the slaves file in spark/conf/slaves
Add the hostnames of the machines on which you would want to start a worker as given below on all the machines in the cluster.

master
slave1
slave2
slave3

To start the cluster
$ cd spark/sbin
$ ./start-all.sh
This should start your cluster.
To check if it has been started correctly, open the web UI on the following address.

http://MasterIP:8080/
All the workers should be listed on it as shown below.



To stop your spark cluster
$ cd spark/sbin
$ ./stop-all.sh

Setting up HDFS
We need to setup a HDFS as Spark can read files from it. Also the scala project jar needs to avaliable at a place accessible by all the workers at any time, a HDFS is the best solution to this.

Download Hadoop on all the nodes in the cluster.
$ wget http://www.trieuvan.com/apache/hadoop/common/hadoop-2.2.0/hadoop-2.2.0.tar.gz
$ sudo tar xwzf hadoop-2.2.0.tar.gz
$ sudo mv hadoop-2.2.0 hadoop
$ sudo chown -R sparkuser:spark hadoop

Next, we move onto changing the configuration files for hadoop. Do this for all the machines in the cluster.
Move to the directory with the configuration files
$ cd hadoop/etc/hadoop
Add the following snippets between `<configuration> ... </configuration>` in the following xml files.
$ vim slaves
Add all the hostnames of the machines on which you want to setup the HDFS as shown below.
master
slave1

slave2
slave3

$ vim core-site.xml

```
<property>

  <name>fs.default.name</name>

  <value>hdfs://master:54310</value>

</property>
```

In place of master you should put the master hostname.

$ vim hdfs-site.xml

```
<property>

  <name>dfs.replication</name>

  <value>4</value>

</property>

<property>

  <name>dfs.namenode.name.dir</name>

  <value>file:/home/madhura/hadoop/hdfs/namenode</value>

</property>

<property>

  <name>dfs.datanode.data.dir</name>

  <value>file:/home/madhura/hadoop/hdfs/datanode</value>

</property>
```

The replication factor should be set to the number of nodes in the HDFS. Here we have 4 nodes.

We need to create directories as specified in hdfs-site.xml under the properties of dfs.namenode.name.dir and dfs.datanode.data.dir on all the machines where we want HDFS.

Move to the hadoop folder
$ cd hadoop
$ mkdir hdfs/datanode
$ mkdir hdfs/namenode
$ sudo chown -R sparkuser:spark hdfs

Now to setup the HDFS we need to run the following command in the master.
$ hadoop namenode -format

To start your HDFS
$ cd hadoop/sbin
$ ./start-dfs.sh

To check if the daemons have started, just try the jps command on the master.
$ jps

31405 NameNode
31796 SecondaryNameNode
31533 DataNode
5816 Jps
On the slaves, the ouput should be as follows.
$ jps

31533 DataNode
5816 Jps

To stop your cluster.
$ cd hadoop/sbin
$ ./stop-dfs.sh

$pwd
/home/sparkuser
$vim .bashrc
Add the following lines in bashrc
#Hadoop variables
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/home/sparkuser/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export SPARK_HADOOP_VERSION=2.2.0
export HADOOP_MAPRED_HOME=$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

$source .bashrc

Steps to run the program :

Unzip the folder GE_Spark and extract all it's contents onto your sparkuser home.
Run these following commands
$chmod +x Setup_env.sh
$./Setup_env.sh

This bash file will start Hadoop and Spark. It will also download some packages required for the execution of the python scripts present in the zip file.

$chmod +x TimeSeriesDemo.sh
$./TimeSeriesDemo.sh

This bash script will generate data sets to test the code on along and will run the spark program without any issues. Enter the input according to the prompts.

The final output will be shown in the form Distance and Location and the end of the execution of this script.

NOTE :
Few changes need to be made in the code before running on your system.
Open SimpleApp.scala present in DTWspark/src/main/scala

1. Line 334: Change the IP to the IP of the spark master
2. Line 401: Change the IP to the IP of the spark master
3. Line 412: Set the required spark executor memory value
4.Line 418 : val sc = new SparkContext("spark://<Master hostname>:7077", "Simple App", "<spark installation directory on master", List("hdfs://hadoopmasterhostname:54310/simple-project_2.10-1.0.jar"))
5.Line 421 : val dRDD = sc.textFile("hdfs://hadoopmasterhostname:54310/inputfilename",10)
Eg: If your input files are named input00, input01, input02, and so on and your hadoop master hostname as master
val dRDD=sc.textFile("hdfs://master:54310/input*",10)

Web UI's to view information about the cluster :

http://<masterIP>:8080 → to view information about the Spark cluster.

http://<masterIP>:50070 → to view information about Hadoop cluster.

http://<masterIp>:4040 → to view information about the running application.