Running Spark on EC2 Documentation

Pseudo code

Input: D vectors data file in format created from Multicore_KMeans, U EC2 master cluster URL, K clusters, I iterations

Step 1: Initialize spark context as standalone cluster at U

Step 2: Create <String>RDD L with the lines in D as a string elements

Step 3: Check if the next element in L is the start of a vector. If true go to Step 4, else go to Step 5. If there are no more elements go to Step 6. --This loop just takes the floating points in the file and adds them to a vector V until the start of a new vectors in the file is recognized, which at that point it will add the current vector V to an RDD R and create a new vector V, and continues until there are no elements left

Step 4: Add vector V to a <Vector>RDD R and creates a new and empty vector V, then go back to Step 3

Step 5: Extract the floating points within the string element from L, then adds each floating point to a vector V, then go back to Step 3.

Step 6: Train a kmeans model based R as a set of vectors using the Spark KMeans algorithm with K number of clusters --spark uses the k-means|| algorithm to initialize the first centroids

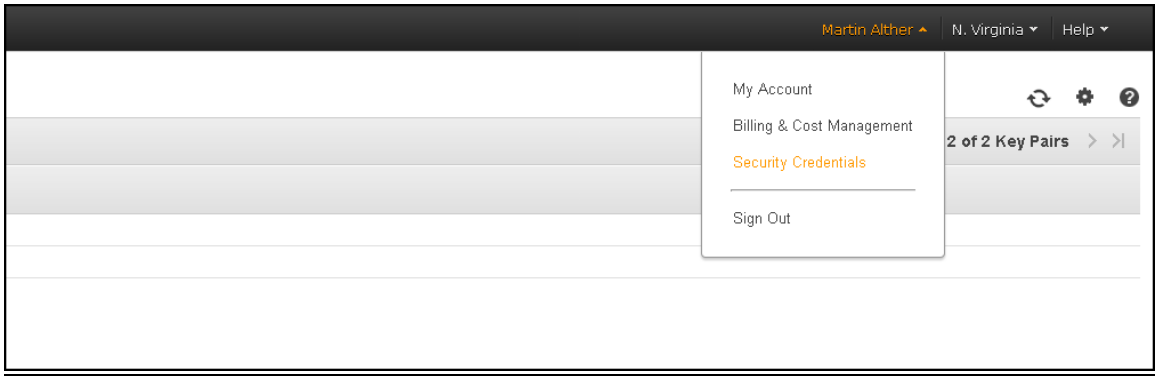
Step 7: Iterate the kmeans algorithm I times or until convergence of centroids

Output: Print the centroids, points, cost, etc.

How to run the code

Preparing to use Amazon AWS

Step 1: Login to the AWS console. Under the account name at the top right, select security credentials.



Step 2: Click on the Access Keys tab and get or create the AWS access key id and AWS secret access key, then save them someplace for later.
Step 3: Under the services tab in the top left, select EC2. Once on EC2 dashboard, go to left side and click on the Network and Security tab and select Key Pairs.
Step 4: Create a new key pair, and save the key pair name as well as the .pem private key file.

Installing Spark

Step 1: Download the latest version of Spark from spark.apache.org/downloads.html
Step 2: Find the version of Scala that your version of Spark uses from either the README.md file in the Spark folder or from http://spark.apache.org/docs/latest
Step 3: Download the version of Scala that Spark is dependent on from http://www.scala-lang.org/download/all.html
Step 4: Download and install the Simple Build Tool(sbt) from http://www.scala-sbt.org/download.html
Step 5: Navigate into the Spark folder and execute the command "./sbt/sbt assembly" to install Spark on the machine.

```
Qwuke@VistaHome /cygdrive/c/Users/Qwuke/Documents/spark-1.0.0/spark-1.0.0
$ sbt/sbt assembly
```

Launching the EC2 cluster

Step 1: Set environment variables for the AWS access key and secret access key that we saved in **Preparing to use AWS Step 2** with the commands:
export AWS_ACCESS_KEY_ID=<Access Key Here>
export AWS_SECRET_ACCESS_KEY=<Secret Access Key Here>

Step 2: In the Spark folder you had downloaded, navigate to the directory named "ec2".
Step 3: Run the "spark-ec2" file with these arguments:
./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> --instance-type=<INSTANCE_TYPE> launch <cluster-name>
Where <keypair> is the name of the key pair we saved in **Preparing to use AWS Step 4**,
<key-file> is the .pem file associated with that generated key pair
<num-slaves> is the number of slave instances to launch with the master instance
<INSTANCE_TYPE> is the type of instance to be launched
and <cluster-name> is the name of the cluster we give it and will work with from now on in the EC2 scripts.

```
Qwuke@VistaHome /cygdrive/c/Users/Qwuke/Documents/spark-1.0.0/spark-1.0.0/ec2
$ spark-ec2 -k MYKEYPAIR -i MYKEYPAIR.pem -s 2 --instance-type=r3.xlarge launch MYSPARKCLUSTER --resume
```

If everything works correctly, you will see the master and slave nodes launching and installing Spark automatically, and at the end it will print the URL of the master node. You may get an error for not setting your .pem private key file permissions to 600, or the launch script may time out after waiting for an instance to initiate. However, you can fix the permissions and run the same launch command from the spark-ec2 file and add the argument '--resume'. This should fix basic launch errors.

Step 4: After the cluster has finished initializing, you can ssh into its root directory with:

./spark-ec2 -k <keypair> -i <key-file> login <cluster-name>

Step 5: To end the EC2 Spark cluster, ssh out of the cluster and use:

./spark-ec2 destroy <cluster-name>

Run this command before deleting anything else in the EC2 dashboard on AWS.

Running code or KMeans on the cluster

Step 1: Use scp or WinSCP with the user "root", the URL of the master node that was printed at the end of the EC2 launch script, and the .pem file to transfer your spark executable(e.g. a KMeans.jar) and data files.

For example if they were in a folder called KMeansDir, this would be the command

```
$ scp -i MYKEYPAIR.pem -r /KMeansDir root@ec2-55-5-55-5-5-55-5.compute-1.amazonaws.com:/
```

Step 2: ssh into the master node using **Step 4 of Launching the EC2 Cluster.** To copy the directory that was scp'ed onto the master node In the previous step, go into the Spark folder installed on the instance and run ~/spark-ec2/copy-dir /KMeansDir to copy the folder to all the slave nodes.

Step 3: In order to pass a data file into a spark executable when it is being run on the master AND the slave nodes, you will have to load the data file into HDFS with the command *ephemeral-hdfs/bin/hadoop fs -put datafile.txt*. This will make the data accessible to the slave nodes as well.

Step 4: Now that everything is ready, pass the spark executable to the spark-submit file in the Spark folder bin. You will also need to pass arguments for its main class and the URL of the master node to spark-submit. Example code:

\bin\spark-submit --class KMeans --master root@ec2-55-5-55-5-5-55-5.compute-1.amazonaws.com KMeansDir\KMeans.jar

Step 5: This will run the Spark executable on all of the nodes. If you wish you pass arguments to the executable itself, simply pass them as normal arguments after listing the location of the executable.

```
$ \bin\spark-submit --class KMeans --master root@ec2-55-5-55-5-5-55-5.compute-1.amazonaws.com KMeansDir\KMe
ans.jar datafile.txt 5 100
```

Step 6: To run the example ConvertedKMeans.jar, use the steps above but pass the class argument to spark-submit as: *--class ConvertKMeans*, and pass these arguments after the .jar file location in this order: *ConvertedKMeans.jar <datafile-name/location> <K-number-of-centroids> <number-of-iterations>*.

Stopping the cluster

Step 1: Go to the Ec2 directory on your local machine from where you launched the cluster, in the terminal.

Step 2: Type the following command in the terminal

$ ./spark-ec2 destroy <your cluster-name>

Cleanup (Important)

Step 1: Logon to Amazon AWS and under Services select 'Ec2'.

Step 2: Under the 'Instances' tab in the left column; click on 'Instances'.

Step 3: Locate all your Hadoop instances and select them. On the top locate 'Actions' drop down button and click 'Stop' to stop the instances. You can start it and connect to the same settings whenever you want. If you terminate it, you have to create a new instance all together.