Exercise 6

More Apache Spark and Python

Prior Knowledge
Unix Command Line Shell
Simple Python

Learning Objectives
Using Spark on EC2
Accessing S3 files on Spark
Reading CSV files in Spark
Seeing the differences between Spark and Hadoop by performing the Wind Analysis in Spark
Spark SQL
Spark statistics

Software Requirements
(see separate document for installation of these)

- Apache Spark 1.5.1
- Python 2.7.x
- Nano text editor or other text editor

Part A. Starting Spark in EC2

1. Do you remember the Access Key and Secret Key from Exercise 1? You need those now.

2. In a terminal window type:
```
export AWS_ACCESS_KEY_ID=<your access key here>
export AWS_SECRET_ACCESS_KEY=<your secret key here>
```

3. Now change into the Spark EC2 directory:
```
cd ~/spark-1.5.1/ec2
```

4. Now let's launch a Spark cluster in EC2. Replace XX with your user details so these match the locations of your key files and so you can identify your own spark cluster

```
./spark-ec2 --key-pair=oxcloXX \
--identity-file=/home/oxclo/oxcloXX.pem \
--region=eu-west-1 \
-s 1 \
launch oxcloXX-spark-cluster
```

The –s 1 indicates that there is just one slave (you could launch more but that might be expensive).

You should see output like:

```
Setting up security groups...
Searching for existing cluster my-spark-cluster in region
eu-west-1...
Spark AMI: ami-1ae0166d
Launching instances...
Launched 1 slave in eu-west-1a, regid = r-52c4f5ff
Launched master in eu-west-1a, regid = r-c2c7f66f
Waiting for AWS to propagate instance metadata...
Waiting for cluster to enter 'ssh-ready' state.........

Warning: SSH connection error. (This could be temporary.)
Host: ec2-52-16-96-164.eu-west-1.compute.amazonaws.com
SSH return code: 255
SSH output: ssh: connect to host ec2-52-16-96-164.eu-
west-1.compute.amazonaws.com port 22: Connection refused
```

5. Maybe go grab a coffee ☺ This takes a while

6. After a while the system will start logging a lot more as the setup on EC2 starts happening.
Eventually you will see:

```
Setting up ganglia
RSYNC'ing /etc/ganglia to slaves...
ec2-52-31-197-16.eu-west-1.compute.amazonaws.com
Shutting down GANGLIA gmond:                                [FAILED]
Starting GANGLIA gmond:                                     [  OK  ]
Shutting down GANGLIA gmond:                                [FAILED]
Starting GANGLIA gmond:                                     [  OK  ]
Connection to ec2-52-31-197-16.eu-west-1.compute.amazonaws.com closed.
Shutting down GANGLIA gmetad:                               [FAILED]
Starting GANGLIA gmetad:                                    [  OK  ]
Stopping httpd:                                             [FAILED]
Starting httpd: httpd: Syntax error on line 154 of /etc/httpd/conf/httpd.conf: Cannot load
/etc/httpd/modules/mod_authz_core.so into server: /etc/httpd/modules/mod_authz_core.so: cannot open
shared object file: No such file or directory
                                                            [FAILED]
[timing] ganglia setup:  00h 00m 02s
Connection to ec2-52-16-96-164.eu-west-1.compute.amazonaws.com closed.
Spark standalone cluster started at http://ec2-52-16-96-164.eu-west-1.compute.amazonaws.com:8080
Ganglia started at http://ec2-52-16-96-164.eu-west-1.compute.amazonaws.com:5080/ganglia
Done!
```

7. It seems in this distribution Ganglia is not working, but let's carry on regardless!

8. Find the Spark URL in the log above and go to that page in your browser. You might want to leave this open as we'll need it later.

9. Let's login to the master:

   ```
   ./spark-ec2 -k oxcloXX -i /home/oxclo/oxcloXX.pem \
   --region eu-west-1 \
   login oxcloXX-spark-cluster
   ```

   You see:

   ```
   Last login: Wed Nov 18 09:27:01 2015 from ip-172-31-9-125.eu-
   west-1.compute.internal

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

   https://aws.amazon.com/amazon-linux-ami/2013.03-release-notes/
   Amazon Linux version 2015.09 is available.
   ```

10. This basically just SSH's you into the master. You could do the same from the EC2 console as before.

11. We are going to need your AWS credentials in this session as well so that spark can access S3 resources. In this session type

    ```
    export AWS_ACCESS_KEY_ID=<your access key here>
    export AWS_SECRET_ACCESS_KEY=<your secret key here>
    ```

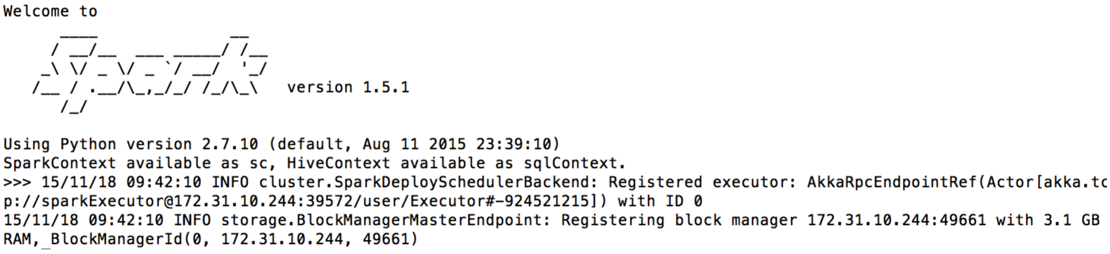
12. Type

    ```
    cd spark
    ```

13. Now start pyspark once again. This time we are going to add in a Spark Package that supports easy reading of CSV files:

    ```
    bin/pyspark \
     --packages com.databricks:spark-csv_2.11:1.2.0
    ```

    You should see:



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.5.1
      /_/

Using Python version 2.7.10 (default, Aug 11 2015 23:39:10)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 15/11/18 09:42:10 INFO cluster.SparkDeploySchedulerBackend: Registered executor: AkkaRpcEndpointRef(Actor[akka.tc
p://sparkExecutor@172.31.10.244:39572/user/Executor#-924521215]) with ID 0
15/11/18 09:42:10 INFO storage.BlockManagerMasterEndpoint: Registering block manager 172.31.10.244:49661 with 3.1 GB
RAM, BlockManagerId(0, 172.31.10.244, 49661)
```

14. We are going to use Spark's SQL support which in turn uses Apache Hive. This combined with the CSV package we saw earlier makes it very easy to work with data.
    First let's tell spark we are using SQL:
    ```
    from pyspark.sql import SQLContext
    sqlContext = SQLContext(sc)
    ```

15. Now let's load the data into a DataFrame.

    ```
    df = sqlContext.\
    read.format('com.databricks.spark.csv').\
    options(header='true', inferschema='true').\
    load('s3n://oxclo-wind/2015/*')
    ```

16. You should see a lot of log go by.

17. The df object we have is not an RDD. Its basically a SQL construct. But we can easily convert it into an RDD.

    ```
    winds = df.rdd
    ```

18. You can see the structure of each row by:

    ```
    winds.first()
    ```

19. Let's do the normal step of mapping the data into a simple <K,V> pair. Each column in the row can be accessed by the syntax e.g. row.Station_ID

    We can therefore map our RDD with the following:

    ```
    mapped = \
    winds.map(lambda s: (s.Station_ID, \
    s.Wind_Velocity_Mtr_Sec))
    ```

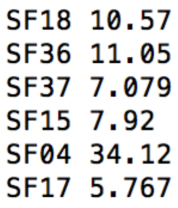
20. We can simply calculate the maximum values with this reducer:

    ```
    maxes = mapped.reduceByKey(lambda a, b: a if (a>b) else b)
    ```

21. And once again collect / print:

    ```
    for (k,v) in maxes.collect(): print k,v
    ```

22. You will see a bunch of log before the following appears:



```
SF18 10.57
SF36 11.05
SF37 7.079
SF15 7.92
SF04 34.12
SF17 5.767
```

PART B – Using SQL

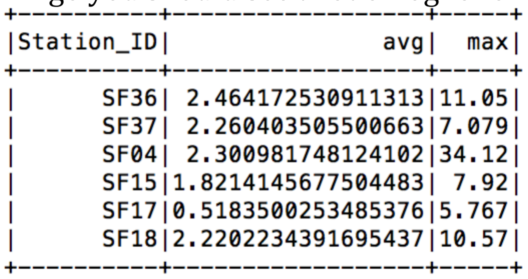
23. There is an easier way to do all this if you are willing to write some SQL.

24. First we need to give our DataFrame a table name:
```
df.registerTempTable('wind')
```

25. Now we can use a simple SQL statement against our data.
ALL ON ONE Line type:

```
sqlContext.sql("SELECT Station_ID, avg(Wind_Velocity_Mtr_Sec) as
avg,max(Wind_Velocity_Mtr_Sec) as max from wind group by
Station_ID").show()
```

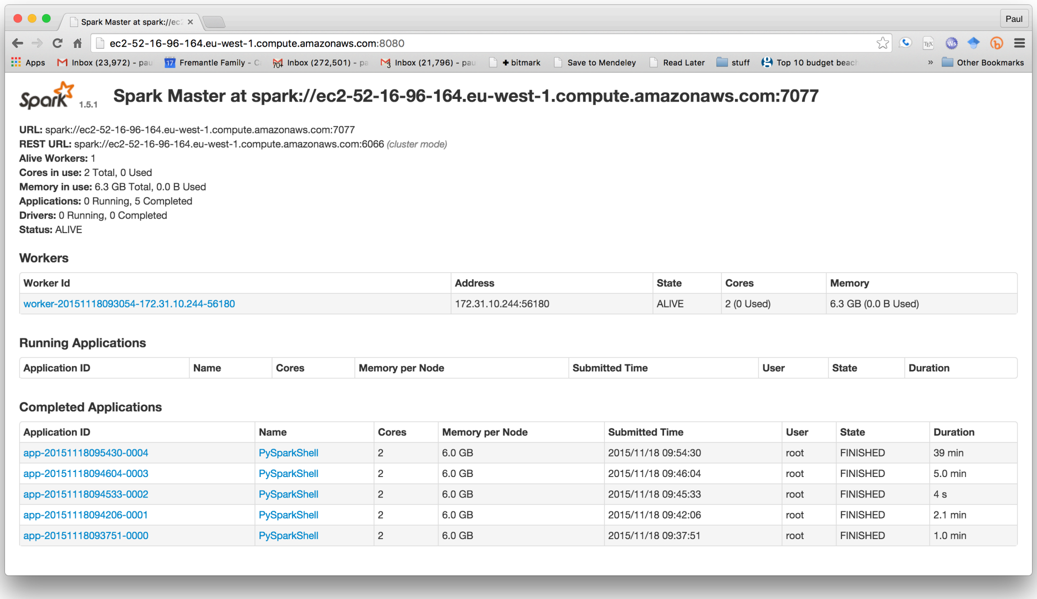
26. Bingo you should see a lot of log followed by:



```
+----------+------------------+-----+
|Station_ID|               avg|  max|
+----------+------------------+-----+
|      SF36| 2.464172530911313|11.05|
|      SF37| 2.260403505500663|7.079|
|      SF04| 2.300981748124102|34.12|
|      SF15|1.8214145677504483| 7.92|
|      SF17|0.5183500253485376|5.767|
|      SF18|2.2202234391695437|10.57|
+----------+------------------+-----+
```

27. Recap. We have:
    a. Started Spark in EC2
    b. Loaded data from S3
    c. Used SQL to read in CSV files
    d. Explored Map/Reduce on those CSV files
    e. Used SQL to query the data.

28. Go back to the browser view of the Spark console and you can take a look at the jobs that have been run:



Spark Master at spark://ec2-52-16-96-164.eu-west-1.compute.amazonaws.com:7077

URL: spark://ec2-52-16-96-164.eu-west-1.compute.amazonaws.com:7077
REST URL: spark://ec2-52-16-96-164.eu-west-1.compute.amazonaws.com:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 6.3 GB Total, 0.0 B Used
Applications: 0 Running, 5 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20151118093054-172.31.10.244-56180	172.31.10.244:56180	ALIVE	2 (0 Used)	6.3 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20151118095430-0004	PySparkShell	2	6.0 GB	2015/11/18 09:54:30	root	FINISHED	39 min
app-20151118094604-0003	PySparkShell	2	6.0 GB	2015/11/18 09:46:04	root	FINISHED	5.0 min
app-20151118094533-0002	PySparkShell	2	6.0 GB	2015/11/18 09:45:33	root	FINISHED	4 s
app-20151118094206-0001	PySparkShell	2	6.0 GB	2015/11/18 09:42:06	root	FINISHED	2.1 min
app-20151118093751-0000	PySparkShell	2	6.0 GB	2015/11/18 09:37:51	root	FINISHED	1.0 min

29. Quit the pyspark shell:

```
quit()
```

30. Exit the SSH session:

```
exit
```

31. We must remember to stop our cluster as well (its costing money!) From Ubuntu terminal where you started the Spark cluster

```
./ec2-spark –region eu-west-1 destroy oxcloXX-spark-cluster
```

32. Congratulations!