amunategui@gmail.com
amunategui.github.io

A Look at HDFS and JSON

Hadoop HDFS is included in the Spark builds we've been using. To get familiar with it, we'll upload a JSON file to the cluster's HDFS and access it from RStudio.

Most of us using Spark probably don't need to worry about managing an HDFS cluster. Just like most of us don't have to worry about managing a large production database, it is still a good idea to know how to reach it, upload data into it and connect to it from RStudio.

HDFS is part of the Apache Hadoop framework:

> ***Hadoop Distributed File System (HDFS)*** *– a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;*
>
> *(https://en.wikipedia.org/wiki/Apache_Hadoop)*

HDFS can break down big data into blocks and store them on hundreds, even thousands of machines. It offers redundancy, scalability and affordability (by using run-of-the-mill computers to store the data).

Starting our Cluster

We'll start by firing up a small Spark cluster. This time we'll use m1.small (see https://aws.amazon.com/ec2/instance-types/ for the different EC2 instance types).

As per previous sections, log into the **AWS** console and start up your launching instance. Once the light is green, get the SSH connection string, fire up a terminal and log into it.

Change drive to (adjust for your Spark version):

```
cd /home/ec2-user/spark-1.5.2-bin-hadoop2.6/ec2
```

and check that your **.pem** file is there. Also, paste both of your AWS keys into the terminal:

```
export AWS_ACCESS_KEY_ID=your-key-here
export AWS_SECRET_ACCESS_KEY= your-key-here
```

Now you are ready and fire up this cluster (replace with your key -k, identity –I and region -r):

```
./spark-ec2 -k udemy -i udemy.pem -r us-west-2 -s 1 -t m1.small launch --copy-aws-credentials
my-spark-cluster
```

Accessing HDFS on our Master Cluster

Once the clusters are up and running, access the master cluster by copying the last http address:

```
Ganglia started at http://ec2-54-188-155-220.us-west-
2.compute.amazonaws.com:5080/ganglia
Done!
```

Into the following (replace all pertinent bits with your own information):

```
ssh -i "udemy.pem" root@ec2-54-188-155-220.us-west-2.compute.amazonaws.com
```

Now that we're on the master's cluster, we're going to access our temporary HDFS:

The spark-ec2 script already sets up a HDFS instance for you. It's installed in /root/ephemeral-hdfs, and can be accessed using the bin/hadoop script in that directory. Note that the data in this HDFS goes away when you stop and restart a machine.

(https://spark.apache.org/docs/0.6.0/ec2-scripts.html)

As per the help file, change drive to *ephemeral-hdfs*:

```
cd  /root/ephemeral-hdfs
```

And get familiar with HDFS' help:

```
bin/hadoop fs -help
```

We're going to upload the example file 'people.json' that comes with spark to our HDFS directory but first we need to download it:

```
wget https://raw.githubusercontent.com/apache/spark/master/examples/src/main/resources/people.json
```

Then create a new directory in

```
bin/hadoop fs -mkdir jsondir
```

Transfer the file into it:

```
bin/hadoop fs -put people.json jsondir
```

Finally, let's take a peek at the file in its new home:

```
bin/hadoop fs -ls jsondir
```

That's it! You just uploaded a file into HDFS!!

Accessing People.json from RStuido

Create the user account for RStudio while you're still in the terminal:

```
adduser yourname
passwd yourname
…
```

Now we access RStudio from the browser. A quick way to find out the public IP address without going through the AWS console:

```
wget http://ipinfo.io/ip -qO -
```

Then enter it into the browser with the extension 8787:

http://54.188.155.220:8787/

Querying our JSON File

```
library("SparkR", lib.loc="/root/spark/R/lib")
Sys.setenv(SPARK_HOME="/root/spark")

sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)

# here we enter the hdfs path to our json folder we created
people <- read.df(sqlContext,
"hdfs://ec2-54-188-155-220.us-west-2.compute.amazonaws.com:9000
/user/root/jsondir/people.json", "json")

head(people)

# register out json file as a table so we can query it
registerTempTable(people, "people_json_table")
names <- sql(sqlContext, "SELECT distinct name FROM
people_json_table WHERE age <= 50")

head(names)
##    name
## 1   Andy
## 2 Justin

# who is the oldest?
head( sql(sqlContext, "SELECT name, age FROM people_json_table
ORDER BY age DESC"))

##     name age
## 1    Andy  30
## 2  Justin  19
## 3 Michael  NA

# only show the oldest
head( sql(sqlContext, "SELECT name, age FROM people_json_table
ORDER BY age DESC LIMIT 1"))
##     name age
## 1    Andy  30
```

Finally

Don’t forget to **terminate** all your clusters and stop (or **terminate**) your launching instance!