Spark Tutorial (AWS)

Junghoon Kang

1 Overview

In this tutorial, you will learn how to run a Spark application on a cluster of AWS EC2 machines.

2 AWS Components

Before we talk about how to deploy a Spark application to AWS EC2, there are few things that need to be understood.

- IAM
  - Video: http://aws.amazon.com/iam/
  - Documentation: http://docs.aws.amazon.com/IAM/latest/UserGuide/iam-ug.pdf
- EC2
  - Video: http://aws.amazon.com/ec2/
  - Documentation: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf
- S3
  - Video: http://aws.amazon.com/s3/
  - Documentation: http://docs.aws.amazon.com/AmazonS3/latest/gsg/s3-gsg.pdf

Try the following steps before you move on to the next section.

1. Create an IAM user for yourself and log into AWS Console with the IAM user/password.
2. Instantiate a EC2 machine.
3. SSH into the machine.

The step-by-step tutorial is written in the above documents.

3 Launch Spark Cluster

3.1 EC2 Key Pair

Create an EC2 key pair though AWS Console so that you could SSH into a master or worker instances in a Spark cluster after you launch the cluster. When the private key is downloaded to your local machine, set the permissions for the private key file to `400` and move the file to `.ssh` directory.

```
$ sudo chmod 400 jung-keypair-uswest2.pem
$ mv jung-keypair-uswest2.pem ~/.ssh
```

3.2 AWS Access Keys

Create an AWS access key pair through AWS Console and set the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` in your local machine.

```
export AWS_ACCESS_KEY_ID=ABCDE1234567890
export AWS_SECRET_ACCESS_KEY=AaBbCcDdEe1234567890!@#$%^&*()
```

Since these keys are equivalent to the ID/password pair of your AWS account, make sure you do not share it with other people or upload it to a publich git repository.

3.3 Spark EC2 Script

The `spark-ec2` script, which is located inside `spark-1.6.0-bin-hadoop2.6/ec2/` directory in your local machine, helps you to launch, manage, and terminate Spark clusters running on Amazon EC2. It automatically sets up a cluster with Spark and HDFS for you.
For example, you can launch a Spark cluster with 3 worker nodes and 1 master node like below:

```
$ spark-ec2 \
--key-pair=jung-keypair-uswest2 \
--identity-file=/home/jung/.ssh/jung-keypair-uswest2.pem \
--region=us-east-1 \
--instance-type=t2.micro \
--slaves=3 \
launch sparkcluster
```

Let me briefly go over each of the options used in this example:

- `--key-pair=KEY_PAIR`
  It specifies which key pair to use on instances. This is the name of EC2 key pair that you have created through AWS console.
- `--identity-file=IDENTITY_FILE`
  It specifies which SSH private key file to use (located in your local machine) for logging into instances.
- `--region=REGION`
  It specifies the region in which to launch EC2 instances. This region should be the same as the region where you have created your EC2 key pair.

  - US East (N. Virginia) : us-east-1
  - US West (N. California) : us-west-1
  - US West (Oregon) : us-west-2

- `--instance-type=INSTANCE_TYPE`
  It specifies the type of instance to launch (default: `m1.large`, which has 2 cores and 7.5 GB RAM). The list of instance types and its cost are provided here:
  http://aws.amazon.com/ec2/instance-types/

- `--slaves=SLAVES`
  It specifies the number of worker nodes to launch (default: 1).

- `sparkcluster`
  The name of your EC2 cluster.

For more information, type:

```
$ spark-ec2 --help
```

After you launch a Spark cluster, you can monitor the instances through AWS console.

4 Spark Application

4.1 Write

Let's use the same application that we have used in **Spark Tutorial (Local Machine)** but with few changes.

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

object WordCount {
    def main(args: Array[String]) {
        val MASTER_ADDRESS = "ec2-52-37-117-184.us-west-2.compute.amazonaws.com"
        val SPARK_MASTER = "spark://" + MASTER_ADDRESS + ":7077"
        val HDFS_MASTER = "hdfs://" + MASTER_ADDRESS + ":9000"
        val INPUT_DIR = HDFS_MASTER + "/wordcount/input"
        val OUTPUT_DIR = HDFS_MASTER + "/wordcount/output"

        val conf = new SparkConf()
            .setMaster(SPARK_MASTER)
            .setAppName("WordCount")

        val sc = new SparkContext(conf)

        // Read in the input text file.
        // Then, for each line in the text file, apply remove_punctuation() function.
        val lines_rdd = sc
            .textFile(INPUT_DIR + "/input.txt", 8) // pass in your own input file
            .map(remove_punctuation)
```

```
        // For each data (line string) in lines_rdd, split it into words.
        // Then, filter out empty strings.
        val words_rdd = lines_rdd
            .flatMap( line => line.split("\\s+") )
            .filter( word => word != "" )

        // For each data (word string) in words_rdd, create a (word,1) tuple.
        // Then, count the number of occurrences for each word.
        val wordcounts_rdd = words_rdd
            .map( word => (word, 1) )
            .reduceByKey( (a, b) => (a + b) )

        // Print the top 15 words which occurs the most.
        wordcounts_rdd
            .saveAsTextFile(OUTPUT_DIR)
    }
    def remove_punctuation(line: String): String = {
        line.toLowerCase
            .replaceAll("""[\p{Punct}]""", " ")
            .replaceAll("""[^a-zA-Z]""", " ")
    }
}
```

Let's go over the changes made from the local machine version of WordCount program:

- `MASTER_ADDRESS`: It is the public DNS of the master node, which could be found in the AWS console after launching a cluster.
- `HDFS_MASTER`: It is the master node's access point to the Hadoop File System in the cluster.
- `INPUT_DIR`: Our application will retrieve the input file from this directory.
- `OUTPUT_DIR`: And our application will write the output to this directory.

4.2 Compile

Create an uber JAR using:

```
$ sbt package
```

4.3 Deploy

First, copy the uber JAR and `input.txt` files into the master node using `scp` command. For example:

```
$ scp -i /home/jung/.ssh/jung-keypair-uswest2.pem \
sparkapp_2.11-0.1.jar \
ec2-user@ec2-52-37-117-184.us-west-2.compute.amazonaws.com:/home/ec2-user/
```

Then, `ssh` into the master instance.

```
$ ssh -i /home/jung/.ssh/jung-keypair-uswest2.pem \
ec2-user@ec2-52-37-117-184.us-west-2.compute.amazonaws.com
```

You should be able to locate your JAR and input files in the home directory of the master node.

4.4 Copy Input File to HDFS

Since our application retrieves the input file from HDFS, we need to copy `input.txt` into HDFS.

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -put \
/home/ec2-user/input.txt /wordcount/input/input.txt
```

You can double check by listing the files in `/wordcount/input` directory in HDFS:

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -ls /wordcount/input/
Warning: $HADOOP_HOME is deprecated.

Found 1 items
-rw-r--r-- 3 ec2-user supergroup 65391 2016-03-13 23:01 /wordcount/input/input.txt
```

If you do not want to type the whole path every time to run `hadoop` executable, you can simply add its path to `.bashrc`.

4.5 Event Log

If you want to see event logs even after your application complete, you need to modify the `/root/spark/conf/spark-defaults.conf` file:

```
# add the below two lines
spark.eventLog.enabled  true
spark.eventLog.dir      /home/ec2-user/eventlog
```

The `/home/ec2-user/eventlog` indicates the directory where you would like to store the event logs. Thus, you need to create the directory before you run your application.

```
$ mkdir /home/ec2-user/eventlog
```

4.6 Run

Finally, let's run the WordCount application. Inside the master node, type:

```
$ /root/spark/bin/spark-submit \
--class WordCount \
sparkapp_2.10-1.0.jar
```

You can see the event logs through Spark's web UI at:
ec2-52-37-117-184.us-west-2.compute.amazonaws.com:8080
where `ec2-52-37-117-184.us-west-2.compute.amazonaws.com` is the public DNS of the master node.

4.7 Check Output

If you list the directories in `/wordcount` in HDFS, you will now see the `output` directory.

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -ls /wordcount
Warning: $HADOOP_HOME is deprecated.

Found 2 items
drwxr-xr-x - ec2-user supergroup 0 2016-03-13 23:01 /wordcount/input
drwxr-xr-x - ec2-user supergroup 0 2016-03-14 01:50 /wordcount/output
```

Let's copy the output directory to `/home/ec2-user` directory:

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -get \
    /wordcount/output \
    /home/ec2-user
```

If you list the output directory, you should be able to see 8 output files as we divided the input file into 8 partitions in the code:

```
part-00000  part-00001  part-00002  part-00003  part-00004
part-00005  part-00006  part-00007  _SUCCESS
```

5 Terminate Spark Cluster

To terminate the cluster, in your local machine, type:

```
$ spark-ec2 destroy \
--region=us-west-2 \
<cluster-name>
```

where `<cluster-name>` is `sparkcluster` in our example.

Important: Please terminate the cluster after you are done using it. You are paying for each EC2 instance you use every hour.

6 Useful HDFS Commands

```
/* List the files in / (root directory of HDFS) */
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -ls /

/* Copy /output file in HDFS to /home/ec2-user directory. */
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -get \
    /output /home/ec2-user

/* Remove all files in HDFS. */
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -rmr /*

/* Put foo.txt into / directory in HDFS. */
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -put \
```

```
    /home/ec2-user/foo.txt /

/* Retrieve the size of all data in HDFS. */
$ sudo /root/ephemeral-hdfs/bin/hadoop fs -du -s -h /
```

7 S3

When we terminate a Spark cluster, we lose all the data stored in the cluster, such as `input.txt` and `output` directory. And in this tutorial, we have copied `input.txt` from the local machine to the master node, using `scp` command. It would be tedious to copy the input file over the network everytime you launch a new cluster. Instead, you could upload the input file to AWS S3 and write your Spark program to retrieve the input file from S3 instead of HDFS. I will not go over this on this tutorial, but if you are interested, you could search online and learn how to do it.