Tutorial:
Introduction to Spark MapReduce

Sayed Hadi Hashemi

Last update: September 14, 2015

1 Overview

Welcome

Welcome to the MapReduce tutorial. This tutorial covers the critical skills needed to develop a Spark MapReduce application. It starts with an already deployed Spark environment where students will execute a series of hands-on labs.

Objectives

Upon completing this tutorial, students will be able to:

- Perform basic operations on HDFS
- Develop a simple MapReduce application (Word Count)
- Run MapReduce applications and examine the output
- Make modifications in MapReduce applications

Structure

This guide is designed as a set of step-by-step instructions for hands-on exercises. It teaches you how to operate the MapReduce service in a functional test environment through a series of examples. Exercises should be completed in the order described in the following steps.

1. Download some input datasets
2. Prepare a HDFS and upload dataset on it
3. Develop a simple "**Word Count**" application
4. Run the application on Spark and examine the output
5. Modify the **"Word Count"** application to discard common words
6. Rebuilt, and rerun the modified application
7. Do it yourself modification

2 Requirements

This tutorial is designed to work on the **Hortonworks Sandbox 2.3** virtual machine. You need to have a working HortonWorks Sandbox machine either locally or on the Amazon Web Services.

Also, all assignments are designed based on **JDK 7** (included in the virtual machine).

Please refer to “Tutorial: Run HortonWorks Sandbox 2.3 Locally” or “Tutorial: Run HortonWorks Sandbox 2.3 on AWS” for more information.

3 Setup Virtual Machine

Step 1: Start the virtual machine; then connect to it through the SSH.

Step 2: After successfully logging in, you should see a prompt similar to the following:

```
[root@sandbox ~]#
```

Step 3: Create a new folder for this tutorial:

```
# mkdir mapreduceSpark-tutorial
```

Step 4: Change the current folder:

```
# cd mapreduceSpark-tutorial
```

4 Prepare the HDFS

MapReduce needs all the input and output files to be located on HDFS. In this section, we will first download some text file as the input dataset. Then we create necessary folders on HDFS. Lastly, we copy the dataset to the HDFS.

For the input dataset some masterpieces by William Shakespeare (Hamlet, Macbeth, Othello, and Romeo & Juliet) will be downloaded directly from the Gutenberg Project.

Step 1: Create a folder for the input datasets:

```
# mkdir dataset
```

Step 2: Download the following Shakespeare books:

```
# wget -c http://www.gutenberg.lib.md.us/2/2/6/2264/2264.txt -P dataset
# wget -c http://www.gutenberg.lib.md.us/2/2/6/2265/2265.txt -P dataset
# wget -c http://www.gutenberg.lib.md.us/2/2/6/2266/2266.txt -P dataset
# wget -c http://www.gutenberg.lib.md.us/2/2/6/2267/2267.txt -P dataset
```

Step 3: Create some folders on HDFS:

```
# hadoop fs -mkdir -p /tutorial/input
```

Step 4: Upload the Dataset to HDFS

```
# hadoop fs -put ./dataset/* /tutorial/input
```

5 Hello World (Word Count)

The Word Count application is the canonical example in MapReduce. It is a straightforward application of MapReduce, and MapReduce can handle word count extremely efficiently. In this particular example, we will be doing a word count within Shakespeare's books.

If you are new to Python programming language, take a look at:

https://pythonspot.com

To learn more about developing MapReduce applications in Python, visit:

http://spark.apache.org/docs/latest/quick-start.html

Step 1: Run the application using the following command and wait for it to be done:

```
# spark-submit ./wordcount.py /tutorial/input/ /tutorial/output
```

Step 2: The output of the application will be on the HDFS. To see the output, you may use the following command:

```
# hadoop fs -cat /tutorial/output/*
```

Step 3 (Optional): The output files can be downloaded from HDFS using the following command:

```
# hadoop fs -get /tutorial/output/* .
```

Step 4 (Optional: Just in case of runtime errors: If you encounter any errors (possibly due to typos), you need to first edit the source file and fix the problems.
Additionally, the output folder on HDFS should be cleaned before a new run using following command:

```
# hadoop fs -rm -r -f /tutorial/output
```

Finally, run the code again.

6 Do it your self

One problem with the current output is that it is too long. In this section we will develop a new application which only shows the top 10 most commonly used words.

Step 1: Try to run `topwords.py` yourself.

> Remember to shut down the virtual machine after you are done.

7 Questions to Think About

1. Why data should be on HDFS?
2. Why should the “output” folder be removed each time?