February 26 2019

Lab 2

Course: Big Data and Machine Learning

Instructors:
Salman Toor, Salman.Toor@it.uu.se

Task-1: Basics of the Hadoop Distributed Filesystem (HDFS)

Hands-on:

This task is based on HDFS, a file system designed for the Hadoop framework. For this task, boot a new instance by selecting the “Hadoop-Node-Snapshot” snapshot. It is a specialized image containing a complete Hadoop framework, configured for running in “pseudo-distributed mode”
(see https://hadoop.apache.org/docs/r2.7.0/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation)

Use “ssc.medium” flavor to start your new instance.

Note:
Set correct hostname and IP address in “/etc/hosts” and “/etc/hostname” file.

The following steps will start all the necessary Hadoop services:

1. Run the following script:

```
$ /hadoop/bin/hadoop namenode -format
```

2. Start Hadoop services:

```
$ cd /hadoop/sbin/
$ ./start-dfs.sh
$ ./start-yarn.sh
```

the script will ask you to add the fingerprints:

Are you sure you want to continue connecting (yes/no)? say “yes”.

3. Check the status of the services:

```
$ jps
```

The Hadoop framework is written in JAVA, and all the services running in Hadoop can be seen by using the “jps” (Java process status) command. The output of the command should include the following services:

NameNode
DataNode
ResourcesManager
SecondaryNameNode
NodeManager

Run following commands:

```
$ /hadoop/bin/hdfs dfs -ls /
```

```
$ /hadoop/bin/hdfs dfs -mkdir /test-data
```

```
$ /hadoop/bin/hdfs dfs -ls /
```

Questions:

1. Explain the difference between NameNode and DataNode?
2. What are the advantages of using HDFS?
3. Explain the last three commands?

Task-2 (Word Count Example)

Hands-on:

This task requires successful execution of a basic Hadoop program. The Hadoop framework executes programs using MapReduce. Now the task is to first download the data files, then upload it to the HDFS and run the word-count application. The code to be run is available as a test example shipped with the Hadoop libraries.

URLs of the Data files:

```
http://www.gutenberg.org/ebooks/20417.txt.utf-8
http://www.gutenberg.org/ebooks/5000.txt.utf-8
```

Tip: On Linux you can use the ‘wget’ command to download data over http.

```
$ /hadoop/bin/hdfs dfs -put <LOCAL-INPUT-DIR> <HDFS-DIR>

$ /hadoop/bin/hdfs dfs -ls /data
```

When data is staged into HDFS, issue the following command to execute the application:

```
$ /hadoop/bin/hadoop jar
/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.7.3.jar wordcount <HDFS-INPUT-DIR>
<HDFS-OUTPUT-DIR>
```

Question:

1. Explain the command(s) you have used to upload the data files in HDFS?
2. Briefly explain the console's output and the results of the word count application (look at the result file produced by the Hadoop job).
3. Explain the application's output?

Task-3 (Calculate π using Hadoop)

Hands-on:

In this task, we will again use the already available example code in Hadoop to calculate π. The details regarding the implementation are available at following link:

http://stevesu.github.io/blog/2014/10/21/how-does-hadoop-calculate-pi/

```
$ /hadoop/bin/hadoop jar
/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.7.3.jar pi <NUMBER-OF-MAPPERS> <NUMBER-OF-POINTS>
```

Question:

1. Explain the command you have used to run the PI example?
2. Briefly explain the implementation available in the above-mentioned link.

Task-4 (Hadoop program in Python)

Hands-on:

This example is from the following website, with a few changes:
http://rare-chiller-615.appspot.com/mr1.html

First you need to insall python:

```
$ sudo apt-get install python-minimal
```

1. Prepare data

In this example we use part of the data from Centers for Disease Control and Prevention. The columns are:

- ID
- Gender
- Age
- Weight(lbs)
- "Do you consider yourself now to be overweight(=1), underweight(=2), or about the right weight(=3)?
- Would you like to weigh more(=1), less(=2), or stay(=3) about the same?

You can download data from the following link:

```
$ wget  https://www.dropbox.com/s/9jkjs6benu0pejn/CDC.csv
```

The data looks like this:

```
'000'00003', 1, 21, 72, 180, 3, 1
'00004', 2, 32, 63, 135, 1, 2
'00034', 2, 42, 63, 128, 1, 2
03', 1, 21, 72, 180, 3, 1
'00004', 2, 32, 63, 135, 1, 2
'00034', 2, 42, 63, 128, 1, 2
```

2. Upload CSV file to HDFS

First, create a folder in HDFS called "lab2" by using "hdfs dfs -mkdir" command. After that, using "put" command, upload local CSV file to HDFS like the following

```
$ /hadoop/bin/hdfs dfs -mkdir /lab2
$ /hadoop/bin/hdfs dfs -put CDC.csv /lab2/CDC.csv
```

3. Write Mapper and Reducer in Python

The example explains how to calculate average age for each gender.
Let's start from Mapper.

In Hadoop, there is a Java program called Hadoop streaming-jar. This program internally read (stdin) and print out (stdout) line by line. Therefore, Python can read each line as a string and parse it by using functions like strip and split(",").
For example, the first line would be parsed like this.

```
"'00003', 1, 21, 72, 180, 3, 1"  -> ['00003', "1", "21", "72",
"180", "3", "1"]
```

So, you need to only pick the second element (gender) and the third element (age) in each array, and print out (stdout) them as Key-Value pair.

Here is Mapper.py example.

Note: Use an editor such as *nano* or *vim* to write your code.

Mapper.py

```
#!/usr/bin/python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    line = line.strip()
    line = line.split(",")

    if len(line) >=2:
        sex = line[1]
        age = line[2]

        print '%s\t%s' % (sex, age)
```

Once the mapper code is ready, you need to prepare the reducer. In reducer, you parse string coming from mapper.py as Key-Value pair, and keep them in dictionary like { gender : [age,.....] }. Once you keep the data in dictionary, you can easily get average.

Here is the reducer.py example.

Reducer.py

```
#!/usr/bin/python
#Reducer.py
import sys

sex_age = {}

#Partitoner
for line in sys.stdin:
    line = line.strip()
    sex, age = line.split('\t')

    if sex in sex_age:
        sex_age[sex].append(int(age))
    else:
        sex_age[sex] = []
        sex_age[sex].append(int(age))

#Reducer
for sex in sex_age.keys():
    ave_age = sum(sex_age[sex])*1.0 / len(sex_age[sex])
    print '%s\t%s'% (sex, ave_age)
```

4. Change access permission on Mapper and Reducer

Please change access permission on your mapper and reducer as below.

```
$ chmod +x mapper.py
$ chmod +x reducer.py
```

5. Test your mapper and reducer

As you may notice, you can simulate process-flow by just using "cat" command in your terminal. You are right. You can test your mapper and reducer like this.

```
$ cat CDC.csv | python mapper.py | python reducer.py
1   38.4
2   46.0
```

6. Run your Map Reduce on Hadoop

You should use hadoop-streaming.jar. (the path to hadoop-streaming.jar may be different from my path, so it might be good for you to search where it is in your machine).

In your terminal, move to the directory where you saved mapper.py and reducer.py. Then, execute the following command. You can see that Map Reduce is starting. (I wrote the following command by line for easily showing options, but you need to write the following command in one line.)

```
$ /hadoop/bin/hadoop jar /hadoop/share/hadoop/tools/lib/hadoop-
streaming-2.7.3.jar  -mapper Mapper.py  -reducer Reducer.py  -
input /lab2 -output /lab2-output  -file Mapper.py  -file
Reducer.py
```

7. Get the result

You can see a text file called "part-00000" in the output folder you specified in your hadoop-streaming command.

In the text file, you can find your result.

```
$/hadoop/bin/hadoop fs -cat /lab2/ave_age/part-00000
1     38.4
2     46.0
```

Question:

1. What is Hadoop streaming?
2. Can we use Hadoop streaming with R scripts?
3. How many mappers and reducer have been used while running the task-4? can we change the number of mappers?