Assignment 7 **(20 Oct 2021)**

1 Hadoop Assignment

Prerequisites:

- Python 2/3
- Unix command-line tools like cd, sort
- Basic knowledge of piping, ex: *ls | grep name*
- Working installation of Hadoop[1]

In this assignment we shall make use of the Hadoop streaming api to write our map reduce code. Hadoop Streaming uses Unix standard streams as the interface between Hadoop and your program, so you can use any language that can read standard input and write to standard output to write your MapReduce program. However, we shall use Python for this assignment. All we need to do is write a script for the mapper and reducer, and hadoop will take care of the rest.

First, we shall simulate the behaviour of a map reduce program using simple unix commands to understand how the mapper and reducer works in Section 2.1.

2.1 Map Reduce Simulation

For this simulation we shall do a simple word count. The problem statement is: Given a text file (https://norvig.com/big.txt), get the count of every word in it. Now, the first order of business is to write the mapper.

2.1.1 Word Count Mapper

The job of the mapper is simple, read lines from `stdin` and spit out the key value pairs to `stdout` . Write a python script for the same. An example of expected output from the input is given in Table 2.1.1. Make sure to include the python shebang in your scripts and `chmod +x yourscript.py` . Test your script by running `cat inputfile | ./mapper.py`

Input File
~~hello world~~
testing testing
hello

Mapper Output
hello,1
world,1
testing,1
testing,1
hello,1

Table 1: Mapper Example

[1]Hadoop Installation Guide for Ubuntu 16.04: `https://tinyurl.com/hhe9f4f`

2.1.2 Word Count Reducer

The next task is to write the script for the reducer. The reducer job is to take the output of the mapper and spit out the final count of every word to `stdout`. We will simulate the sorting phase of the hadoop frame work with `sort` com- mand, so that all the same words are ordered together. Test your script by
`catinputfile | ./mapper.py | sort -t ',' -k1 | ./reducer.py`

Mapper Output	hello,1	**Reducer Output**
hello,1	**sort**	
world,1	hello,1	hello,2
testing,1	testing,1	world,1
testing,1	testing,1	testing,2
hello,1	world,1	

Table 2: Reducer Example

```
#!/usr/bin/python

"""mapper.py"""


importsys

# input comes from STDIN (standard input)

forlineinsys.stdin:
    # remove leading and trailing whitespace

    line=line.strip()
    # split the line into words

    words=line.split()
    # increase counters
    forwordinwords:

        # write the results to STDOUT (standard output);
        # what we output here will be the input for the #
        Reduce step, i.e. the input for reducer.py

        #

        # comma delimited; the trivial word count is 1

        print(f'{word},1')
```

```
#!/usr/bin/env python
"""reducer.py"""

importsys

current_word=None current_count=0

#This loop will only work when the input #to
the script is sorted forlineinsys.stdin:

    #read line and split by comma

    #recall, we used comma as    delimiter in mapper

    line=line.strip().split(',')

    #get the key and val, in this case #word
    is the key and count is the val
    word,count=line[0],int(line[1])


    ifcurrent_word==None:      #initialie
        current_word=word
        current_count=count
    elifcurrent_word==word:      #increment the count

        current_count+=count else:
    #spit current word and

        print(f'{current_word},{current_count}') current_word=word
        current_count=count

#spit last word

print(f'{current_word},{count}')
```

2.1.3 Run it in Hadoop

Now that we have written our mapper and reducer, we are ready to execute our program in Hadoop.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar¥
  -input path/to/inputfile¥
  -output path/to/outputdir¥
  -mapper path/to/mapper.py¥
```

```
-reducer reducer.py
```

2.2 Hadoop Assignment

Now that you have learnt how a basic map reduce program works, solve the following.

1. Implement a map reduce program to find all distinct words in the file. Per- form data cleaning in the mapper such that all punctuation are removed and all words are lowercased.

inputfile	**Map Reduce output**
Hello World! Apache hadoop. apache spark.	apache hadoop hello spark world

Table 3: Distinct Words MR example

2. Extend the word count example to include a combiner. Simply use `-combiner combiner.py` option.
3. You are given a dataset of N points and you are given the C candidate points. Implement a map reduce program to assign each of the N points to the nearest (Euclidean distance) candidate point and update the candidate points by taking the average of all the points that were assigned to it. You may hard code the candidate points in your mapper if you want. Make use of the iris[2] dataset, which is in the format

sepallength, sepalwidth, petallength, petalwidth, class

Use the following three points given in Table 4 as candidate points.For this exercise you may use consider the sepal length and sepal width, and remove the rest.

5.8,4.0,1.2,0.2,Iris-setosa
6.1,2.8,4.0,1.3,Iris-versicolor
6.3,2.7,4.9,1.8,Iris-virginica
Table 4: Candidate Points

HINT: creating the multi-key,val pairs in mapper like $<< N_i, C_i >, 1 >$ may be useful

[2] http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Ready to use Dockerfile to create an image with Hadoop already set up. or you can use the steps to set up hadoop on your own machine. Save the following as in a file named Dockerfile, and run `sudo docker build .`

```
apt-get update
apt-get install default-jdk wget -y apt-
get install python3 -y
wget
    http://mirrors.estointernet.in/apache/hadoop/common/hadoop-2.10.0/hadoop-2.10.0.tar.gz
tar -xzvf hadoop-2.10.0.tar.gz
ENVJAVA_HOME $(readlink -f /usr/bin/java | sed "s:bin/java::") RUNmv
hadoop-2.10.0 /usr/local/hadoop
ENVPATH /usr/local/hadoop/bin:$PATH
rm -rf hadoop-2*
```