Cloud Computing (CS351), Aug-Dec 2020

Assugnment 6-7

Due Date: 14/15 Oct 2020

1 Hello Docker!

In this assignment you will get a taste of Docker. Docker is an immensely powerful virtualization tool. Learning its use can save you a lot of time. How? Lets think of this scenario.

You are a software engineer working for the startup Moby Corps. Ltd. Due to a sad frame of events, one of your co-workers is hospitalized and you need to fill in. Sadly its the weekend, and you want to get out of the office as soon as possible. But your boss Mr. Moby Dock, will fire you if you don't complete the work.

Your co-worker was working on a piece of python code that should be work Python 3.8. Your task is to finish implementing the the functions in his code. You don't have time to set up another python environment. So just usedocker to pull a container that is already set up and ready to go! Simply follow the steps below. Depending on your system, you might have to use `sudo` for all commands given below.

1. Install Docker Community Edition on your machine : `https://docs.docker.com/install/`
   On ubuntu it should be as easy as
   ```
   apt-get install docker docker.io -y
   ```
2. type **docker** in your terminal and check if its properly working.
3. Go to docker `https://hub.docker.com`. Search for the Python official image. And pull the image having python version 3.8.
   ```
   docker pull python:3.8-slim
   #check the downloaded image
   docker images
   ```
4. Fire up a container container.
   ```
   #you can name it anything you want
   docker run -dit --name=pyContainer python:3.8-slim
   ```

```
#your container should be listed here
docker container ls
#go inside your container, remember this
dockerexec-it pyContainer /bin/bash
```

If you have done it correctly, the something like the following should show up in your terminal

```
root@O2UwU3:/#
```

5. Inside the container go ahead check of the version python that you want is running correctly.

```
root@O2UwU3:/# python -c"print('hello world')"
root@O2UwU3:/# python --version
#take note of the directory structure
root@O2UwW3:/# ls
# exit the container like so
root@O2UwW3:/#exit
```

6. The container is basically running linux, so think of it as being inside the terminal of your own laptop. You can run **apt-get update** and see for yourself. However, it is running in isolation, so its interaction with your machine is limited (can be changed). This container will have python3.8 installed by default, so that you don't needlessly waste time setting up a another installation of python. This concept extends to any tool that you want to use!

7. Since the container is running isolated and running inside your terminal, the only way for you now is to create a new file inside the container and use commandline tools like vim, nano to make write any sort of script inside the container. This could be an annoyance.(if you are okay with it,then good for you.) Lets fix that. Since we cannot edit a launched container, we will stop and delete it and create a new container where we mount a folder of your local machine inside the container, so that any changes you make in that directory of your local machine shows up in your container. So, in your local terminal, run the following

```
# recall what you named your container
# if you dont, just 'docker container ls'
docker container stop pyContainer
docker container rm pyContainer
#create a new folder or use an existing one
mkdir testfolder
#find out the path of your folder
pwd
# should print out something like /Users/userName/
docker run -dit --name=pyC¥
```

```
        -v /Users/userName/testfolder:/myfolder¥
        python:3.8-slim
    dockerexec-it pyC /bin/bash
    root@O2UwU3:/# ls
    # your mounted folder should up as 'myfolder'. Any changes in this
    # directory should show up in both your machine and this container
```

8. So just pull in your co-workers file, make the necessary changes and make sure it runs. And that it! You are done!

Here is the code that you need to finish implementing

```
fromtypingimportCallable, List
importmath
#DO NOT MAKE UNNECESSARY CHANGES
classDistanceFuncs:
    defcalc_distance(
        self, point_a: List[float], point_b: List[float], dist_func: Callable,/
    )->float:
        """ Calculates distance between two points using the passed dist_func """
        returndist_func(point_a, point_b)

    @staticmethod
    defeuclidean(point_a: List[float], point_b: List[float],/)->float:
        """
        Calculates Euclidean Distance between two points Example:
        >>> DistanceFuncs.euclidean([1,1],[1,1])
        0.0
        """
        returnmath.dist(point_a, point_b)
    @staticmethod
    defmanhattan_distance(point_a: List[float], point_b: List[float],/): """Compute
        the manhattan_distance between two points""" raiseNotImplementedError()
    @staticmethod
    defjaccard_distance(point_a: List[float], point_b: List[float],/): """Compute
        the jaccard_distance between two points""" raiseNotImplementedError()
defmain():
    """Demonstrate the usage of DistanceFuncs """
    pass
if_name_=="_main_":
    main()
```

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

hello,1

sort

hello,1
testing,1
testing,1
world,1

Reducer Output
hello,2
world,1
testing,2

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 .`

```
FROMubuntu:16.04

RUNapt-get update
RUNapt-get install default-jdk wget -y
RUNapt-get install python3 -y
RUNwget
    http://mirrors.estointernet.in/apache/hadoop/common/hadoop-2.10.0/hadoop-2.10.0.tar.gz RUNtar -
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
RUNrm -rf hadoop-2*
```