Hadoop Streaming Using Python – Word Count Problem

Hadoop Streaming is a feature that comes with Hadoop and allows users or developers to use various different languages for writing MapReduce programs like Python, C++, Ruby, etc. It supports all the languages that can read from standard input and write to standard output. We will be implementing Python with Hadoop Streaming and will observe how it works. We will implement the word count problem in python to understand Hadoop Streaming. We will be creating ***mapper.py*** and ***reducer.py*** to perform map and reduce tasks.

Let's create one file which contains multiple words that we can count.

Step 1: Create a file with the name *word_count_data.txt* and add some data to it.

```
cd Documents/                                    # to change the
directory to /Documents
touch word_count_data.txt                 # touch is used to
create an empty file
nano word_count_data.txt                 # nano is a command line
editor to edit the file
cat word_count_data.txt                          # cat is used to
see the content of the file
```

```
dikshant@dikshant-Inspiron-5567:~/Documents$ touch word_count_data.txt
dikshant@dikshant-Inspiron-5567:~/Documents$ nano word_count_data.txt
dikshant@dikshant-Inspiron-5567:~/Documents$ cat word_count_data.txt
geeks for geeks is best online coding platform
welcome to geeks for geeks hadoop streaming tutorial
dikshant@dikshant-Inspiron-5567:~/Documents$
```

Step 2: Create a **mapper.py** file that implements the mapper logic. It will read the data from STDIN and will split the lines into words, and will generate an output of each word with its individual count.

```
cd Documents/                                    # to change the
directory to /Documents
touch mapper.py                    # touch is used to create an
empty file
cat mapper.py                    # cat is used to see the
content of the file
```

Copy the below code to the *mapper.py* file.

Python3

```
#!/usr/bin/env python

# import sys because we need to read and write data to STDIN and STDOUT
import sys
```

```
# reading entire line from STDIN (standard input)
for line in sys.stdin:
    # to remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()

    # we are looping over the words array and printing the word
    # with the count of 1 to the STDOUT
    for word in words:
        # 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
        print '%s\t%s' % (word, 1)
```

Here in the above program **#!** is known as shebang and used for interpreting the script. The file will be run using the command we are specifying.

```
dikshant@dikshant-Inspiron-5567:~$  cd Documents/
dikshant@dikshant-Inspiron-5567:~/Documents$ cat mapper.py
#!/usr/bin/env python

# import sys because we need to read and write data to STDIN and STDOUT
import sys
# reading entire line from STDIN (standard input)
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
dikshant@dikshant-Inspiron-5567:~/Documents$
```

Let's test our **mapper.py** locally that it is working fine or not.

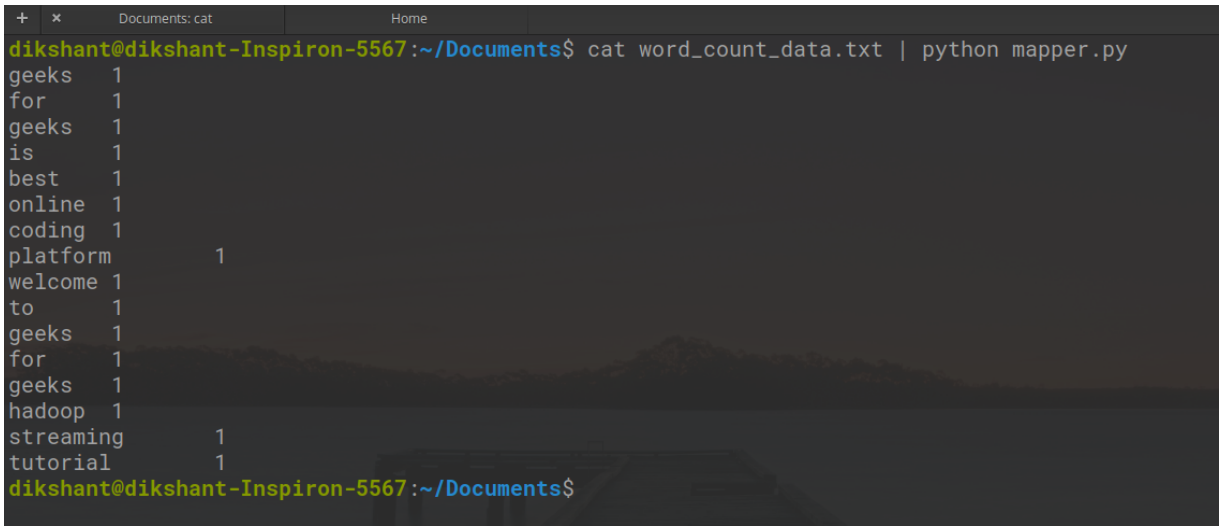
Syntax:

```
cat <text_data_file> | python <mapper_code_python_file>
```

Command(in my case)

```
cat word_count_data.txt | python mapper.py
```

The output of the mapper is shown below.



Step 3: Create a ***reducer.py*** file that implements the reducer logic. It will read the output of mapper.py from STDIN(standard input) and will aggregate the occurrence of each word and will write the final output to STDOUT.

```
cd Documents/                                    # to change the
directory to /Documents
touch reducer.py                          # touch is used to create
an empty file
```

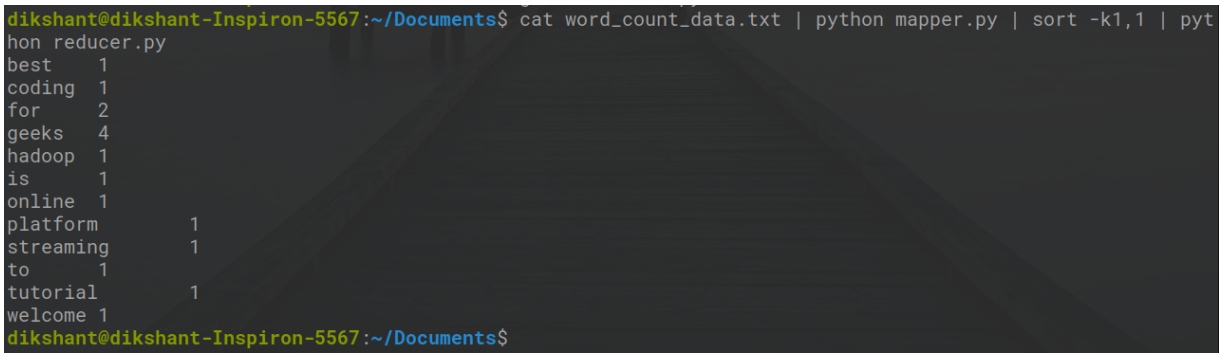
Python3

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # splitting the data on the basis of tab we have provided in mapper.p
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
```

```
    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Now let's check our reducer code reducer.py with mapper.py is it working properly or not with the help of the below command.

```
cat word_count_data.txt | python mapper.py | sort -k1,1 | python
reducer.py
```

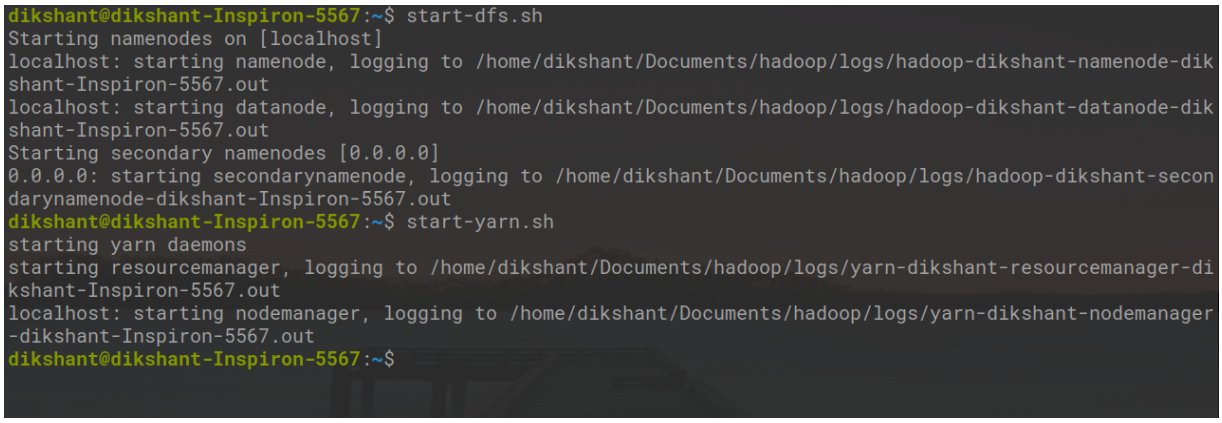


We can see that our reducer is also working fine in our local system.

Step 4: Now let's start all our Hadoop daemons with the below command.

```
start-dfs.sh
```

```
start-yarn.sh
```



Now make a directory **word_count_in_python** in our HDFS in the root directory that will store our **word_count_data.txt** file with the below command.

```
hdfs dfs –mkdir /word_count_in_python
```

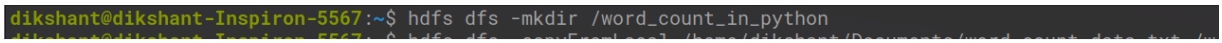
Copy **word_count_data.txt** to this folder in our HDFS with help of copyFromLocal command.

Syntax to copy a file from your local file system to the HDFS is given below:

```
hdfs dfs –copyFromLocal /path 1 /path 2 .... /path n
/destination
```

Actual command(in my case)

```
hdfs dfs –copyFromLocal
/home/dikshant/Documents/word_count_data.txt
/word_count_in_python
```



Now our data file has been sent to HDFS successfully. we can check whether it sends or not by using the below command or by manually visiting our HDFS.

```
hdfs dfs –ls /       # list down content of the root directory

hdfs dfs –ls /word_count_in_python    # list down content of
/word_count_in_python directory
```

Let's give executable permission to our **mapper.py** and **reducer.py** with the help of below command.

```
cd Documents/

chmod 777 mapper.py reducer.py     # changing the permission to
read, write, execute for user, group and others
```

In below image,Then we can observe that we have changed the file permission.

Step 5: Now download the latest **hadoop-streaming jar** file from this Link**.** Then place, this Hadoop,-streaming jar file to a place from you can easily access it. In my case, I am placing it to ***/Documents*** folder where **mapper.py** and **reducer.py** file is present.

Now let's run our python files with the help of the Hadoop streaming utility as shown below.

```
hadoop jar /home/dikshant/Documents/hadoop-streaming-2.7.3.jar \

> -input /word_count_in_python/word_count_data.txt \

> -output /word_count_in_python/output \

> -mapper /home/dikshant/Documents/mapper.py \

> -reducer /home/dikshant/Documents/reducer.py
```

In the above command in **-output**, we will specify the location in HDFS where we want our output to be stored. So let's check our output in output file at location ***/word_count_in_python/output/part-00000*** in my case. We can check results by manually vising the location in HDFS or with the help of cat command as shown below.

```
hdfs dfs –cat /word_count_in_python/output/part–00000
```

Basic options that we can use with Hadoop Streaming

Option	Description
-mapper	The command to be run as the mapper
-reducer	The command to be run as the reducer

Option	Description
-input	The DFS input path for the Map step
-output	The DFS output directory for the Reduce step

Last Updated : 19 Jan, 2022

11

Similar Reads

- What is Hadoop Streaming?
- Difference between Hadoop 1 and Hadoop 2
- Difference Between Hadoop 2.x vs Hadoop 3.x
- Hadoop - HDFS (Hadoop Distributed File System)
- Hadoop - Features of Hadoop Which Makes It Popular
- How to Execute Character Count Program in MapReduce Hadoop?
- Sum of even and odd numbers in MapReduce using Cloudera Distributi…
- How to Execute WordCount Program in MapReduce using Cloudera Distributi…
- Snakebite Python Package For Hadoop HDFS
- Hadoop - Python Snakebite CLI Client, Its Usage and Command References

Previous

HBase Model in Hadoop

Next

Apache HIVE - Features And Limitations

Article Contributed By :

dikshantmalidev
dikshantmalidev

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : simmytarika5, sumitgumber28

Article Tags : Hadoop

Improve Article | Report Issue

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships