

ROLL NO: 210701263

Exp. No: 2

Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm

AIM:

To run a basic Word Count MapReduce program.

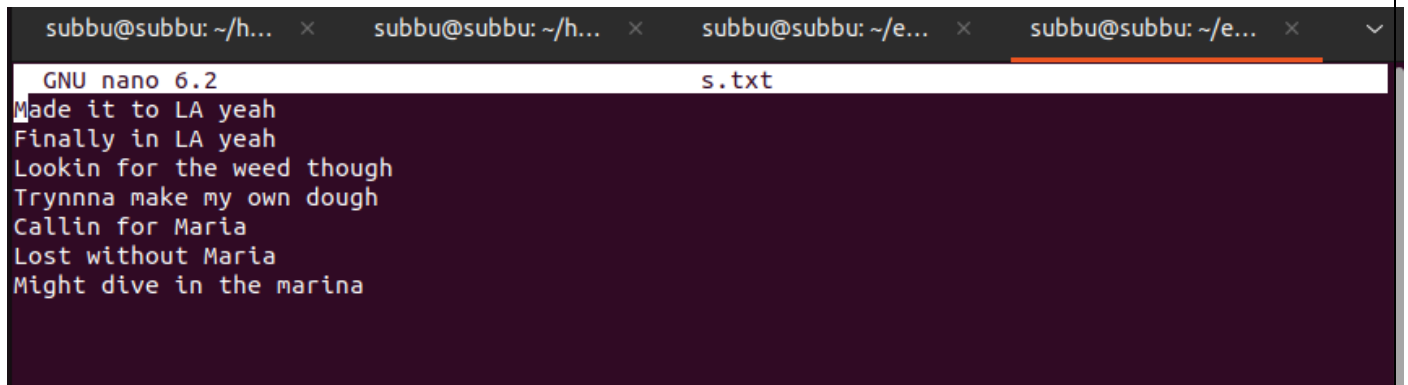
Procedure:

Step 1: Create Data File:

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.
Login with your hadoop user.

nano word_count.txt

Output: Type the below content in word_count.txt



```
subbu@subbu: ~/h... x subbu@subbu: ~/h... x subbu@subbu: ~/e... x subbu@subbu: ~/e... x
GNU nano 6.2 s.txt
Made it to LA yeah
Finally in LA yeah
Lookin for the weed though
Trynnna make my own dough
Callin for Maria
Lost without Maria
Might dive in the marina
```

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

ROLL NO: 210701263

```
nano mapper.py
```

```
# Copy and paste the mapper.py code
```

```
#!/usr/bin/env python3
```

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

```
#!/usr/bin/python3
```

```
import sys
```

```
for line in sys.stdin:
```

```
    line = line.strip() # remove leading and trailing whitespace
```

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

```
    for word in words:
```

```
        print( '%s\t%s' % (word, 1))
```

```
.
```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

```
nano reducer.py
```

```
# Copy and paste the reducer.py code
```

reducer.py

```
#!/usr/bin/python3 from operator
```

```
import itemgetter import sys
```

```
current_word = None current_count
```

```
= 0 word = None for line in
```

```
sys.stdin:     line = line.strip()
```

```
word, count = line.split("\t", 1)
```

```
try:
```

```
    count = int(count)
```

```
except ValueError:
```

```
    continue     if current_word
```

```
== word:         current_count
```

```
+= count     else:
```

```
    if current_word:
```

```
        print( '%s\t%s' % (current_word, current_count))
```

```
current_count = count     current_word = word if
```

```
current_word == word:     print( '%s\t%s' %
```

```
(current_word, current_count))
```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

ROLL NO: 210701263

```
start-all.sh hdfsdfs -mkdir /word_count_in_python hdfsdfs -copyFromLocal  
/path/to/word_count.txt/word_count_in_python
```

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

```
chmod 777 mapper.py reducer.py
```

Step 7: Run Word Count using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

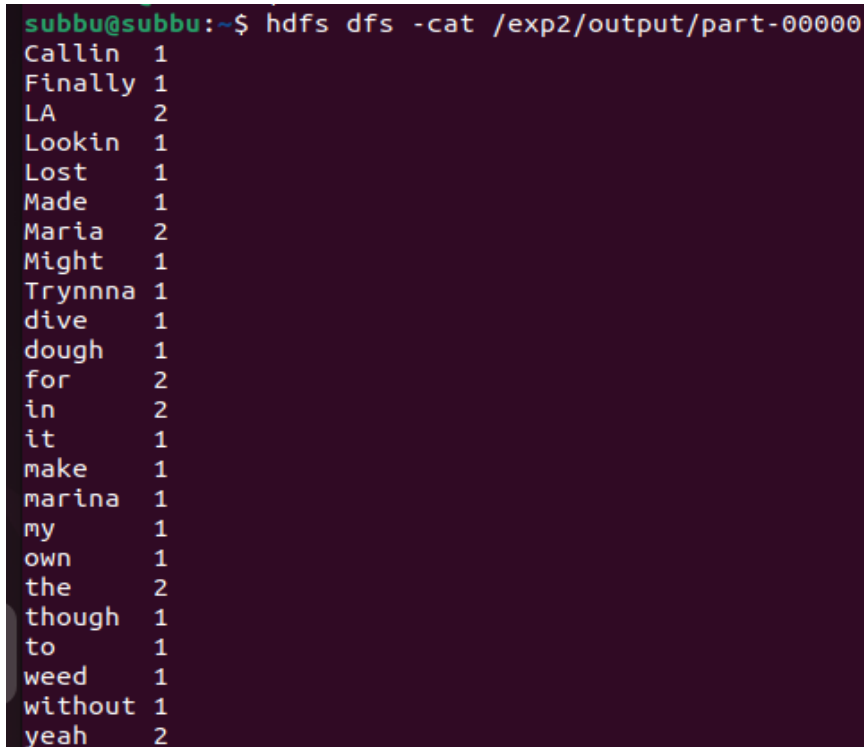
Then run the Word Count program using Hadoop Streaming.

```
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \ -input  
/word_count_in_python/word_count_data.txt \  
-output /word_count_in_python/new_output \  
-mapper /path/to/mapper.py \  
-reducer /path/to/reducer.py
```

Step 8: Check Output:

Check the output of the Word Count program in the specified HDFS output directory.

```
hdfs dfs -cat /word_count_in_python/new_output/part-00000
```



```
subbu@subbu:~$ hdfs dfs -cat /exp2/output/part-00000  
Callin 1  
Finally 1  
LA 2  
Lookin 1  
Lost 1  
Made 1  
Maria 2  
Might 1  
Trynnna 1  
dive 1  
dough 1  
for 2  
in 2  
it 1  
make 1  
marina 1  
my 1  
own 1  
the 2  
though 1  
to 1  
weed 1  
without 1  
yeah 2
```

Result:

Thus, the program for basic Word Count Map Reduce has been executed successfully.

ROLL NO: 210701263