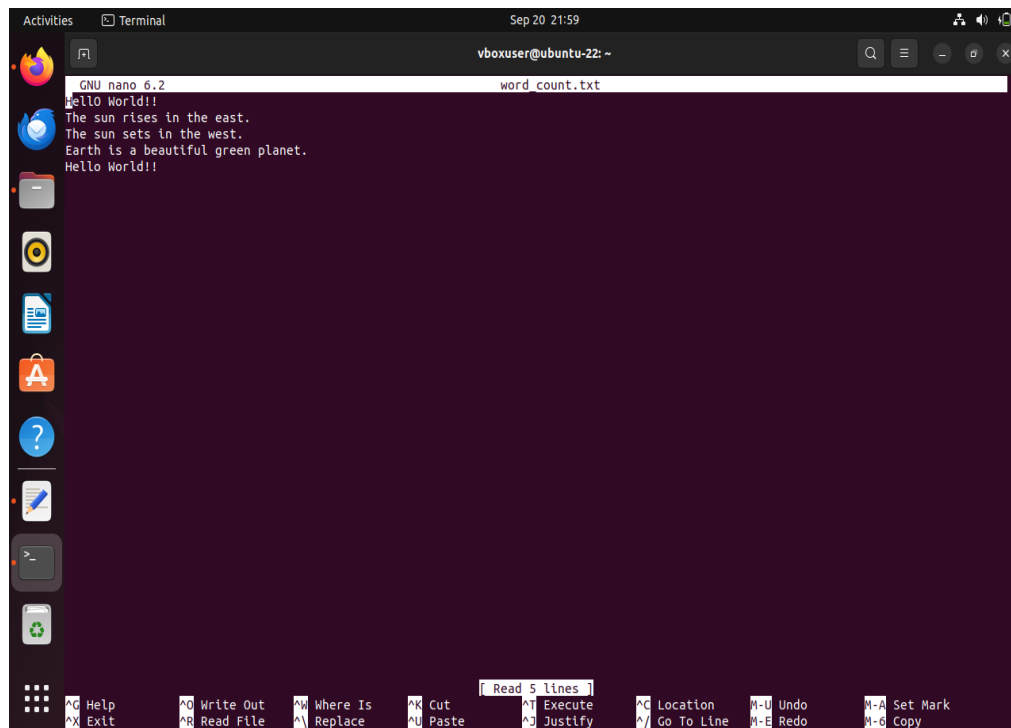


**Exp.No: 2****Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm****AIM:**

To run a basic Word Count MapReduce program.

**PROCEDURE:**

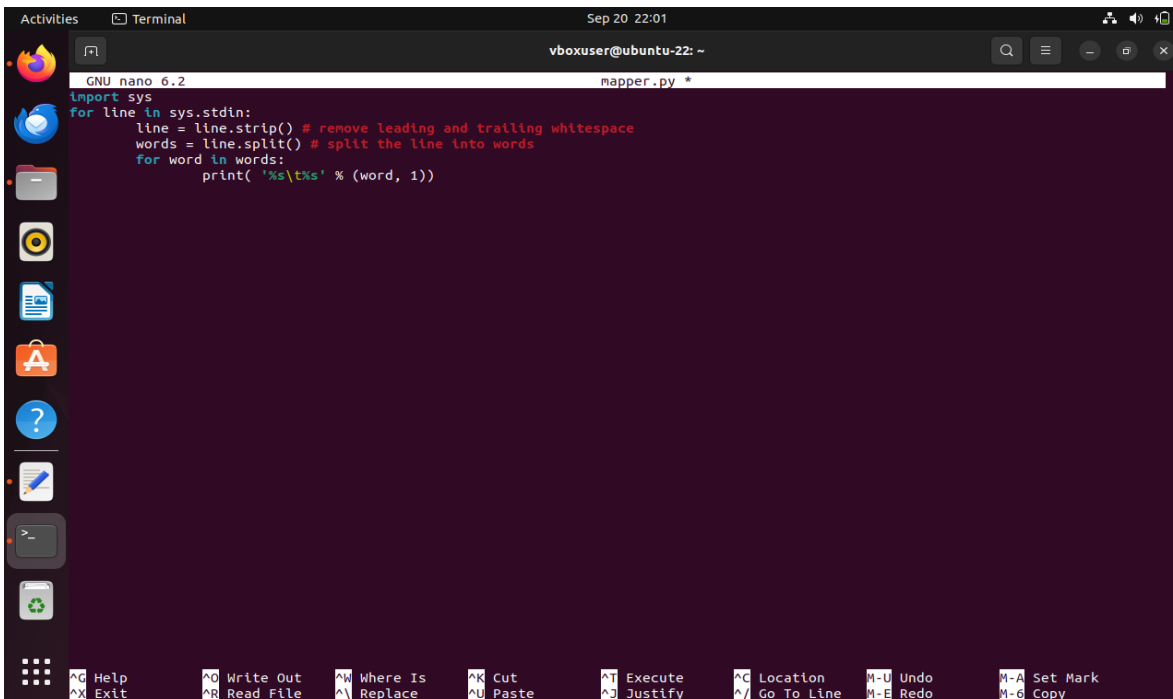
**Step 1: Create Data File:** Create a file named "word\_count\_data.txt" and populate it with text data that you wish to analyse. Login with your hadoop user.

**nano word\_count.txt**

```
GNU nano 6.2 word_count.txt
Hello World!!
The sun rises in the east.
The sun sets in the west.
Earth is a beautiful green planet.
Hello World!!
```

**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.

**nano mapper.py**



```

GNU nano 6.2 mapper.py
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



```

GNU nano 6.2 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.

```
start-all.sh
```

```
hdfs dfs -mkdir /word_count_in_python
```

```
hdfs dfs -copyFromLocal /path/to/word_count.txt/word_count_in_python
```

**Step 5: Make Python Files Executable:** Give executable permissions to your mapper.py and reducer.py files.

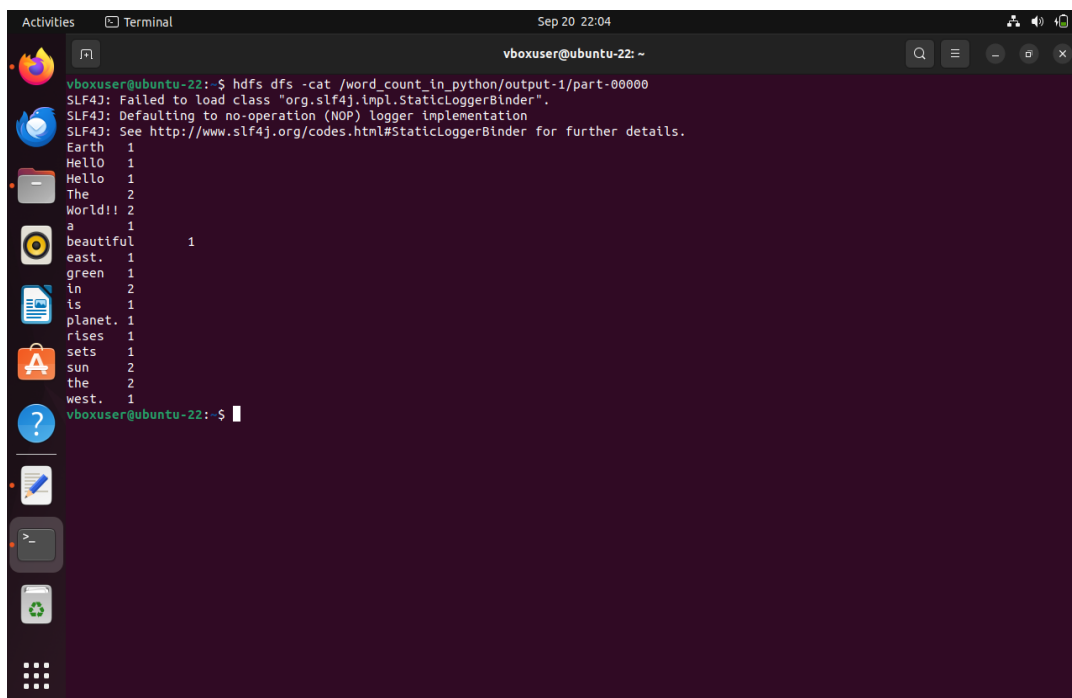
```
chmod 777 mapper.py reducer.py
```

**Step 6: 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 7: Check Output:** Check the output of the Word Count program in the specified HDFS output directory.

```
hdfs dfs -cat /word_count_in_python/output-1/part-00000
```



The screenshot shows a terminal window with the following output:

```
Activities Terminal Sep 20 22:04  
vboxuser@ubuntu-22: ~  
vboxuser@ubuntu-22:~$ hdfs dfs -cat /word_count_in_python/output-1/part-00000  
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  
SLF4J: Defaulting to no-operation (NOP) logger implementation  
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.  
Earth 1  
Hello 1  
Hello 1  
The 2  
World!! 2  
a 1  
beautiful 1  
east. 1  
green 1  
in 2  
is 1  
planet. 1  
rises 1  
sets 1  
sun 2  
the 2  
west. 1  
vboxuser@ubuntu-22:~$
```

**Step-8: Check the Output in the browser.**

localhost:9870/explorer.html#/word\_count\_in\_python/output-1

Block ID: 1073741871  
Block Pool ID: BP-305058674-127.0.1.1-1726117174333  
Generation Stamp: 1047  
Size: 127  
Availability:

- ubuntu-22.4.myguest.virtualbox.org

File contents

Earth	1
Hello	1
Hello	1
The	2
World!!	2
a	1
beautiful	1
east.	1
green	1
in	2
is	1
planet.	1
rises	1
sets	1
sun	2
the	2
west.	1

**RESULT:**

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