

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

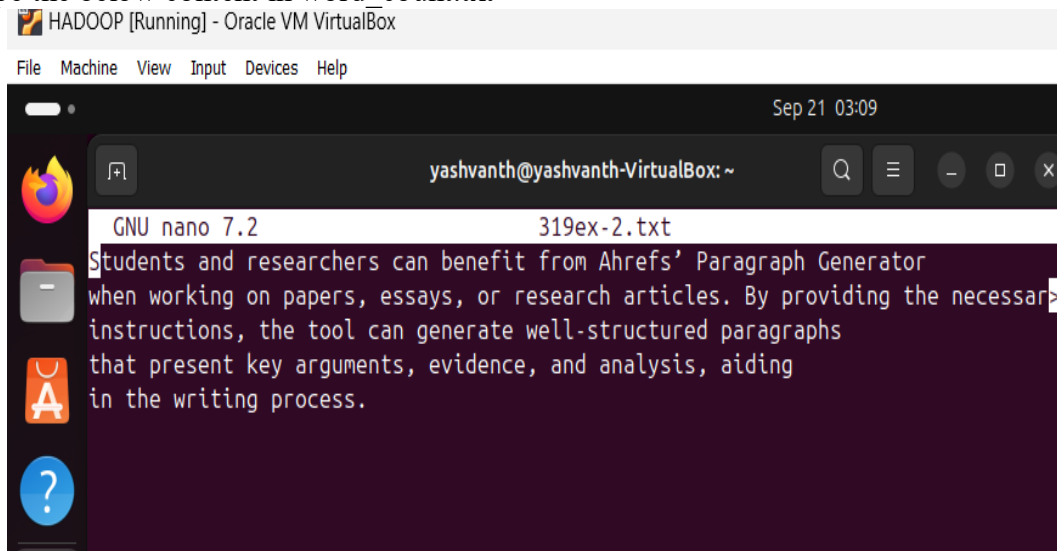
To run a basic Word Count MapReduce program.

**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

**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**

# 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.

```
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.

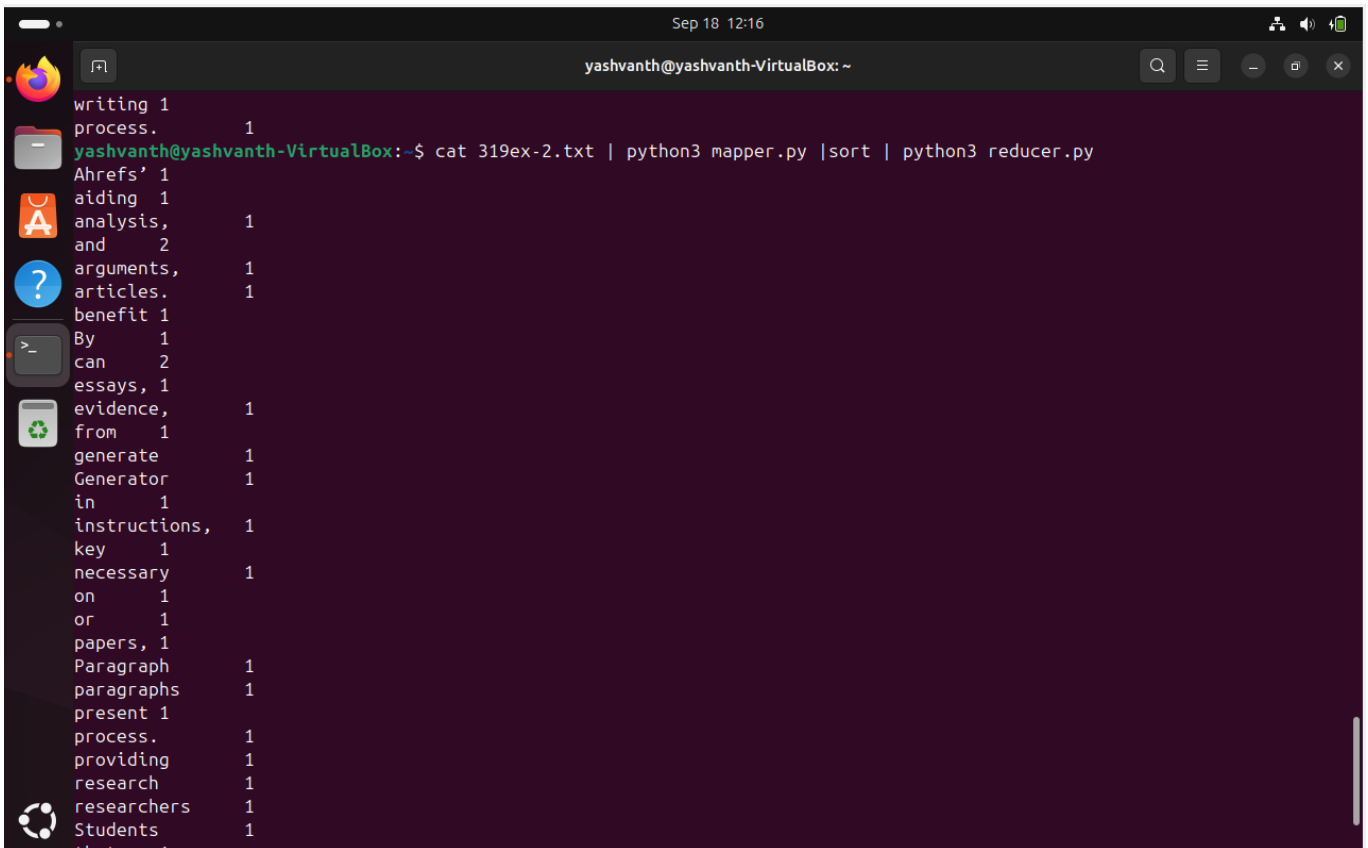
Then run the Word Count program using Hadoop Streaming.

```
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
```



```

writing 1
process. 1
yashvanth@yashvanth-VirtualBox:~$ cat 319ex-2.txt | python3 mapper.py | sort | python3 reducer.py
Ahrefs' 1
aiding 1
analysis, 1
and 2
arguments, 1
articles. 1
benefit 1
By 1
can 2
essays, 1
evidence, 1
from 1
generate 1
Generator 1
in 1
instructions, 1
key 1
necessary 1
on 1
or 1
papers, 1
Paragraph 1
paragraphs 1
present 1
process. 1
providing 1
research 1
researchers 1
Students 1
that 1

```

### Result:

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