

**Exp.No: 2**

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

**AIM:**

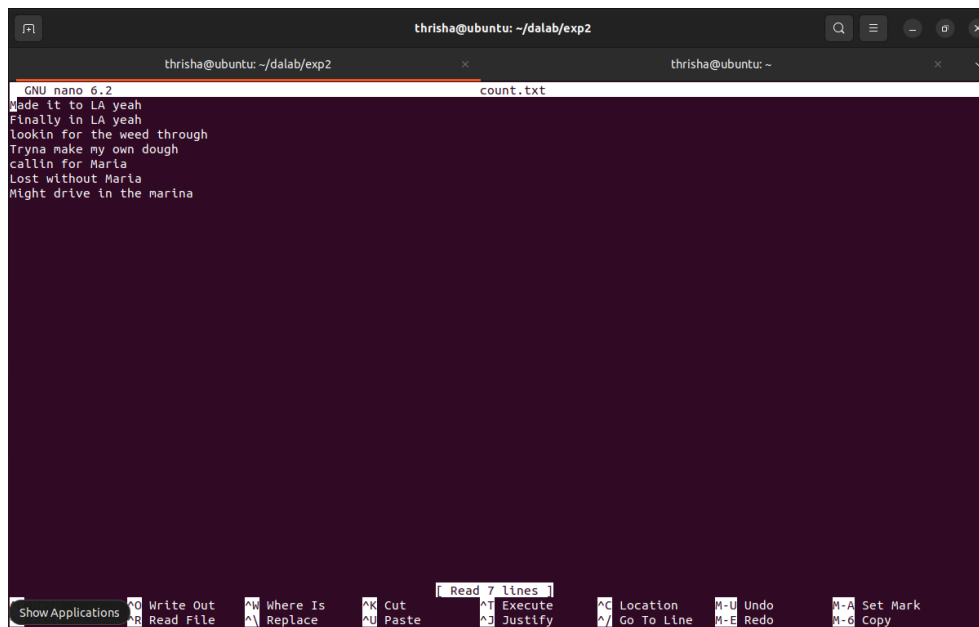
To run a basic Word Count MapReduce program.

**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



```
thrisha@ubuntu: ~/dalab/exp2
GNU nano 6.2 count.txt
made it to LA yeah
Finally in LA yeah
lookin for the weed through
Tryna make my own dough
callin for Maria
lost without Maria
Might drive 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.

```
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.

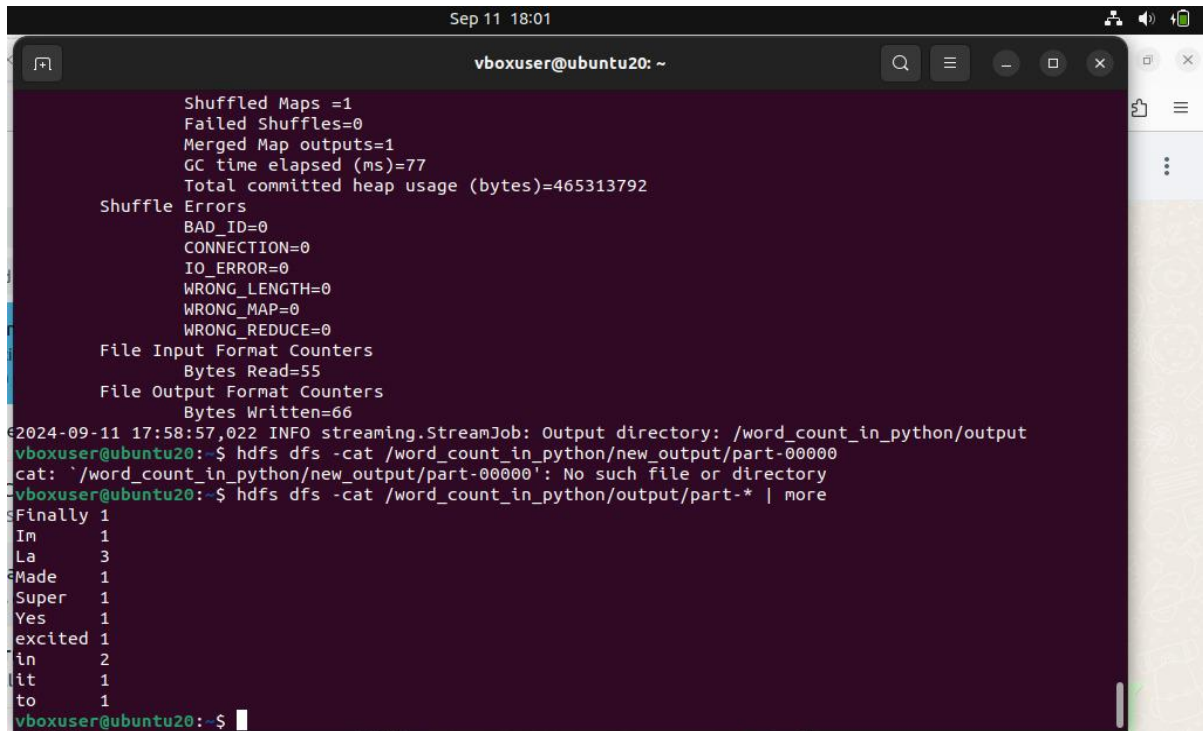
```
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \ -input
/path/to/word_count_in_python/word_count_data.txt \
-output /word_count_in_python/new_output \
-mapper /path/to/mapper.py \
-reducer /path/to/reducer.py
```

```
vboxuser@tamanna:~$ hadoop jar $HADOOP_STREAMING -input /word_count_in_python/word_count.txt -output /word_c
ount_in_python/output -mapper ~/mapper.py -reducer ~/reducer.py
2024-09-21 23:15:19,683 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-21 23:15:19,836 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-21 23:15:19,836 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-21 23:15:19,865 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-21 23:15:20,328 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-21 23:15:20,569 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-21 23:15:20,858 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1585018395_0001
2024-09-21 23:15:20,858 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-21 23:15:21,043 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-21 23:15:21,045 INFO mapreduce.Job: Running job: job_local1585018395_0001
2024-09-21 23:15:21,044 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-21 23:15:21,072 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCo
mmitter
2024-09-21 23:15:21,075 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-21 23:15:21,075 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders
under output directory:false, ignore cleanup failures: false
2024-09-21 23:15:21,214 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-21 23:15:21,216 INFO mapred.LocalJobRunner: Starting task: attempt_local1585018395_0001_m_000000_0
2024-09-21 23:15:21,281 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-21 23:15:21,282 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders
under output directory:false, ignore cleanup failures: false
2024-09-21 23:15:21,333 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2024-09-21 23:15:21,373 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/word_count_in_python/wo
rd_count.txt:0+55
2024-09-21 23:15:21,422 INFO mapred.MapTask: numReduceTasks: 1
2024-09-21 23:15:21,600 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-09-21 23:15:21,600 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-09-21 23:15:21,600 INFO mapred.MapTask: soft limit at 83886080
2024-09-21 23:15:21,600 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-09-21 23:15:21,600 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-09-21 23:15:21,603 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$M
apOutputBuffer
2024-09-21 23:15:21,604 INFO streaming.PipeMapRed: PipeMapRed exec [/home/vboxuser/mapper.py]
2024-09-21 23:15:21,611 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use m
apreduce.task.output.dir
https://chatgpt.com/#
```

### 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
```



The screenshot shows a terminal window titled 'vboxuser@ubuntu20: ~' with a dark purple background. It displays the progress of a Hadoop Word Count job. The output includes statistics for shuffled maps, failed shuffles, merged map outputs, GC time, and heap usage. It also lists shuffle errors (all zero) and file input/output format counters. A log message indicates the output directory is '/word\_count\_in\_python/output'. The user then runs 'hdfs dfs -cat /word\_count\_in\_python/new\_output/part-00000', which returns 'No such file or directory'. Finally, the user runs 'hdfs dfs -cat /word\_count\_in\_python/output/part-\* | more', which displays a list of words and their counts: 'Finally 1', 'Im 1', 'La 3', 'Made 1', 'Super 1', 'Yes 1', 'excited 1', 'in 2', 'lit 1', and 'to 1'.

```
Sep 11 18:01
vboxuser@ubuntu20: ~
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=77
Total committed heap usage (bytes)=465313792
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=55
File Output Format Counters
Bytes Written=66
2024-09-11 17:58:57,022 INFO streaming.StreamJob: Output directory: /word_count_in_python/output
vboxuser@ubuntu20:~$ hdfs dfs -cat /word_count_in_python/new_output/part-00000
cat: '/word_count_in_python/new_output/part-00000': No such file or directory
vboxuser@ubuntu20:~$ hdfs dfs -cat /word_count_in_python/output/part-* | more
Finally 1
Im 1
La 3
Made 1
Super 1
Yes 1
excited 1
in 2
lit 1
to 1
vboxuser@ubuntu20:~$
```

### Result:

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