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.

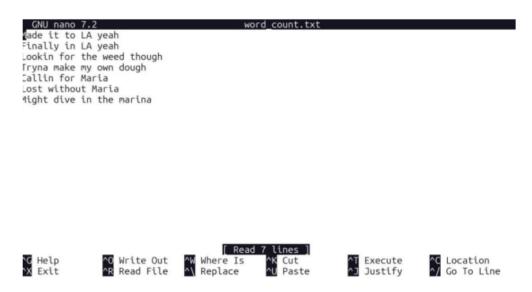
PROCEDURE:

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:



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

CODE:

Mapper.py

#!/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: nano word count.txt
            print( '%s\t%s' % (word, 1))
Step 3: Reducer Logic - reducer.py:
$nano 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
```

```
hadoop@ubuntu:/home ~/wordcount$ hadoop jar /home/hadoop/Documents/hadoop-streaming-3.3.6.jar
                                                                                                                      -inpu
t /word_count_in_python/word_count.txt
                                                 -output /word_count_in_python/output
                                                                                                 -mapper mapper.py
2023-10-25 22:54:51,391 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-10-25 22:54:51,526 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s). 2023-10-25 22:54:51,526 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-10-25 22:54:51,537 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-10-25 22:54:51,785 INFO mapred.FileInputFormat: Total input files to process: 1
2023-10-25 22:54:51,912 INFO mapreduce.JobSubmitter: number of splits:1
2023-10-25 22:54:52,141 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1007878220_0001
2023-10-25 22:54:52,141 INFO mapreduce.JobSubmitter: Executing with tokens: [] 2023-10-25 22:54:52,320 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-10-25 22:54:52,322 INFO mapreduce.Job: Running job: job_local1007878220_0001
2023-10-25 22:54:52,327 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-10-25 22:54:52,328 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCo
2023-10-25 22:54:52,332 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-10-25 22:54:52,333 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders
 under output directory:false, ignore cleanup failures: false
2023-10-25 22:54:52,409 INFO mapred.LocalJobRunner: Waiting for map tasks 2023-10-25 22:54:52,412 INFO mapred.LocalJobRunner: Starting task: attempt_local1007878220_0001_m_000000_0
2023-10-25 22:54:52,455 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-10-25 22:54:52,455 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders
 under output directory:false, ignore cleanup failures: false
2023-10-25 22:54:52,493 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2023-10-25 22:54:52,517 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/word_count_in_python/wo
rd count.txt:0+150
2023-10-25 22:54:52,570 INFO mapred.MapTask: numReduceTasks: 1
```

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/output/part-00000

OUTPUT:

```
hadoop@ubuntu:/home :~/wordcount$ hdfs dfs -cat /word_count_in_python/output/part-00000
Callin 1
Finally 1
LA
Lookin 1
Lost
Made
      1
Maria 2
Might
      1
Tryna
      1
dive
      1
dough
      1
for
       2
in
it
       1
```

RESULT:

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