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.

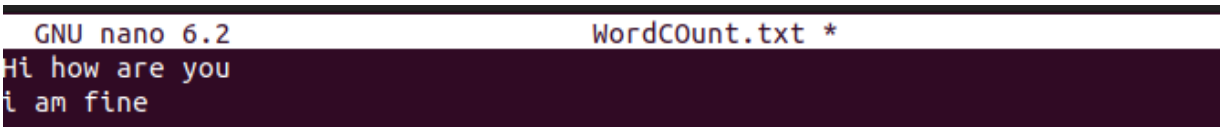
Procedure:

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

```
hadoop@priyav-VirtualBox:~$ hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.
jar -input /word_count_in_python/word_count.txt -output /word_count_in_python/output -mapper mapper.py -r
educer reducer.py
2024-09-02 10:50:26,297 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-02 10:50:26,431 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-02 10:50:26,431 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-02 10:50:26,440 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-02 10:50:26,716 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-02 10:50:26,813 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-02 10:50:26,983 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1711765797_0001
2024-09-02 10:50:26,984 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-02 10:50:27,133 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-02 10:50:27,134 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-02 10:50:27,135 INFO mapreduce.Job: Running job: job_local1711765797_0001
2024-09-02 10:50:27,160 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutpu
tCommitter
2024-09-02 10:50:27,188 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-02 10:50:27,188 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary fold
ers under output directory:false, ignore cleanup failures: false
2024-09-02 10:50:27,262 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-02 10:50:27,264 INFO mapred.LocalJobRunner: Starting task: attempt_local1711765797_0001_m_000000_
0
2024-09-02 10:50:27,316 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-02 10:50:27,320 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary fold
ers under output directory:false, ignore cleanup failures: false
2024-09-02 10:50:27,357 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-09-02 10:50:27,368 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/word_count_in_python
/word_count.txt:0+62
2024-09-02 10:50:27,412 INFO mapred.MapTask: numReduceTasks: 1
2024-09-02 10:50:27,450 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
```

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

```
teju@Ubuntu: ~
teju@Ubuntu:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as teju in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Ubuntu]
2024-09-20 22:07:23,582 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Starting resourcemanager
Starting nodemanagers
teju@Ubuntu:~$ hdfs dfs -cat /WordCount/Output/part-r-00000
2024-09-20 22:09:41,548 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Hi      1
am      1
are     2
fine    2
hi      1
how     1
i       1
you     1
teju@Ubuntu:~$
```

Result:

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