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.

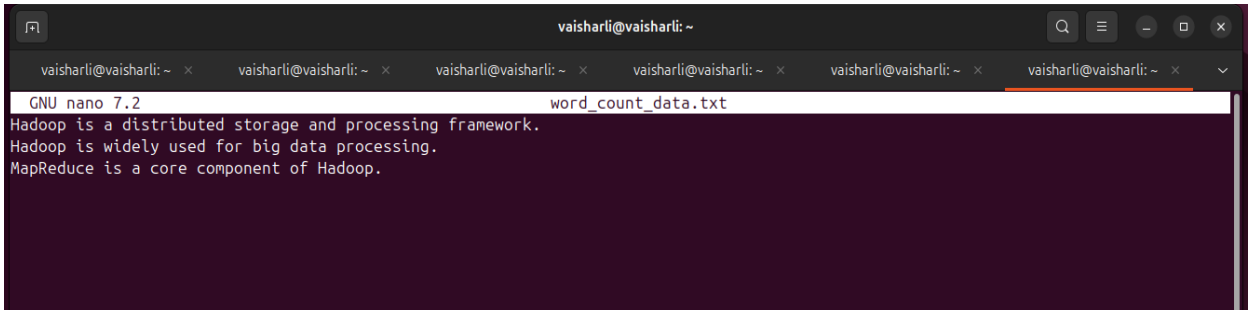
Procedure:

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
```

```
vaisharli@vaisharli:~$ nano mapper.py
vaisharli@vaisharli:~$ jps
5794 ResourceManager
5219 NameNode
5558 SecondaryNameNode
5354 DataNode
5914 NodeManager
8027 Jps
vaisharli@vaisharli:~$ hdfs dfs -mkdir /word_count_in_python
vaisharli@vaisharli:~$ hdfs dfs -copyFromLocal /home/vaisharli/word_count_data.txt /word_count_in_python
vaisharli@vaisharli:~$ chmod 777 mapper.py reducer.py
vaisharli@vaisharli:~$ hadoop jar /home/vaisharli/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
-input /word_count_in_python/word_count_data.txt \
-output /word_count_in_python/output \
-mapper /home/vaisharli/mapper.py \
-reducer /home/vaisharli/reducer.py

2024-09-20 10:49:51,365 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-20 10:49:51,737 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-20 10:49:51,737 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-20 10:49:51,766 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-20 10:49:52,180 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-20 10:49:52,280 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-20 10:49:52,475 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local503562059_0001
2024-09-20 10:49:52,476 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-20 10:49:52,802 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-20 10:49:52,811 INFO mapreduce.Job: Running job: job_local503562059_0001
2024-09-20 10:49:52,812 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-20 10:49:52,846 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCom
itter
2024-09-20 10:49:52,894 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
```

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

```
vaisharli@vaisharli:~$ hdfs dfs -cat /word_count_in_python/output/part-00000
Hadoop  2
Hadoop. 1
MapReduce       1
a       2
and     1
big     1
component       1
core    1
data    1
distributed     1
for     1
framework.      1
is      3
of      1
processing      1
processing.     1
storage 1
used    1
widely  1
vaisharli@vaisharli:~$
```

Result:

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