

Exp.No: 2

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

AIM:

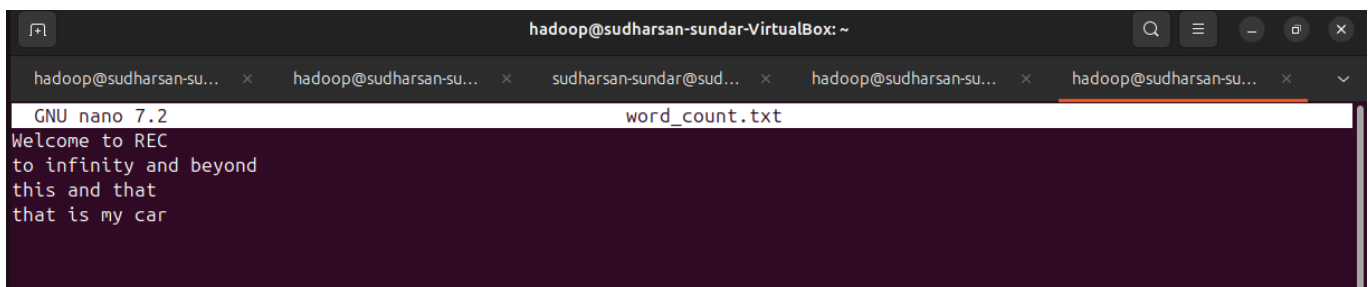
To run a basic Word Count MapReduce program.

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

A screenshot of a terminal window titled 'hadoop@sudharsan-sundar-VirtualBox: ~'. The window shows a nano editor session for 'word_count.txt'. The text inside the file is: 'Welcome to REC', 'to infinity and beyond', 'this and that', and 'that is my car'.**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

```
hadoop@sudharsan-sundar-VirtualBox:~$ hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-input /weatherdata/weather.txt -output /weatherdata/output -file "/home/hadoop/mapper.py" -mapper "python3 mapper.py" -
file "/home/hadoop/reducer.py" -reducer "python3 reducer.py"
2024-09-19 17:00:22,440 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/mapper.py, /home/hadoop/reducer.py] [] /tmp/streamjob5726159080403118813.jar tmpDir=null
2024-09-19 17:00:23,465 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-19 17:00:23,658 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-19 17:00:23,658 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-19 17:00:23,711 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-19 17:00:24,105 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-19 17:00:24,221 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-19 17:00:24,468 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local802387566_0001
2024-09-19 17:00:24,468 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-19 17:00:24,830 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hadoop/mapper.py as file:/tmp/had
oop-hadoop/mapred/local/job_local802387566_0001_cf6d2794-9146-4a6b-bc77-7f42fd109bf9/mapper.py
2024-09-19 17:00:24,882 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hadoop/reducer.py as file:/tmp/had
oop-hadoop/mapred/local/job_local802387566_0001_a308f94d-ee55-4cd1-a1a4-aaf9ff62b356/reducer.py
2024-09-19 17:00:25,048 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-19 17:00:25,050 INFO mapreduce.Job: Running job: job_local802387566_0001
2024-09-19 17:00:25,053 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-19 17:00:25,065 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-09-19 17:00:25,082 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-19 17:00:25,082 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2024-09-19 17:00:25,156 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-19 17:00:25,159 INFO mapred.LocalJobRunner: Starting task: attempt_local802387566_0001_m_0000000_0
2024-09-19 17:00:25,211 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-19 17:00:25,211 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2024-09-19 17:00:25,232 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
```

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
```

```
hadoop@sudharsan-sundar-VirtualBox:~$ hadoop fs -cat /word_count_in_python/output/part-00000
REC      1
Welcome  1
and       2
beyond   1
car       1
infinity      1
is        1
my        1
that      2
this      1
to        2
```

Result:

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