

## EXP 2: Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

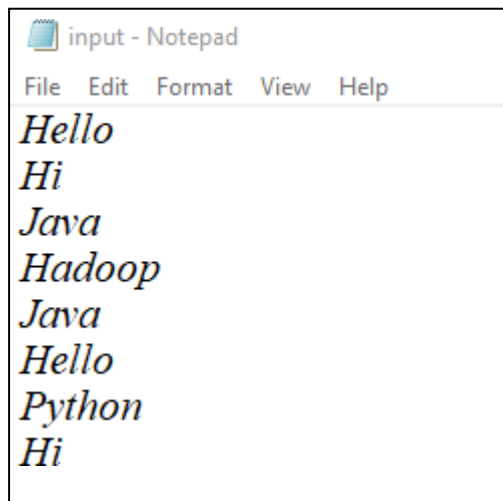
### AIM:

To run a basic Word Count MapReduce program using Hadoop.

### PROCEDURE:

#### Step 1: Create Data File:

Create a file named "input.txt" and populate it with text data that you wish to analyse.



#### 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.

##### mapper.py:

```
#!/C:/Users/user/AppData/Local/Microsoft/WindowsApps/python.exe
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    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.

##### reducer.py:

```
#!/C:/Users/user/AppData/Local/Microsoft/WindowsApps/python.exe
import sys
prev_word = None
prev_count = 0
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t')
    count = int(count)
```

```

if prev_word == word:
    prev_count += count
else:
    if prev_word:
        print('%s\t%s' %(prev_word, prev_count))
        prev_count = count
        prev_word = word
if prev_word == word:
    print('%s\t%s' %(prev_word, prev_count))

```

#### Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data. Run the following commands to store the data in the WordCount Directory.

```

start-all.cmd
cd C:/Hadoop/sbin
hdfs dfs -mkdir /WordCount
hdfs dfs -put C:/Users/user/Documents/DataAnalytics/input.txt /WordCount
hadoop jar C:/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar ^
-input /WordCount/input.txt ^
-output /WordCount/output ^
-mapper "python C:/Users/user/Documents/DataAnalytics/mapper.py" ^
-reducer "python C:/Users/user/Documents/DataAnalytics/reducer.py"

```

#### Step 5: Check Output:

Check the output of the Word Count program in the specified HDFS output directory.

```
hdfs dfs -cat /WordCount/output/part-00000
```

### OUTPUT:

```

C:\> start-all.cmd
C:\> cd C:/Hadoop/sbin
C:\> hdfs dfs -mkdir /WordCount
C:\> hdfs dfs -put C:/Users/user/Documents/DataAnalytics/input.txt /WordCount
C:\> hadoop jar C:/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar ^
-input /WordCount/input.txt ^
-output /WordCount/output ^
-mapper "python C:/Users/user/Documents/DataAnalytics/mapper.py" ^
-reducer "python C:/Users/user/Documents/DataAnalytics/reducer.py"

```

```

C:\> hdfs dfs -cat /WordCount/output/part-00000

```

```

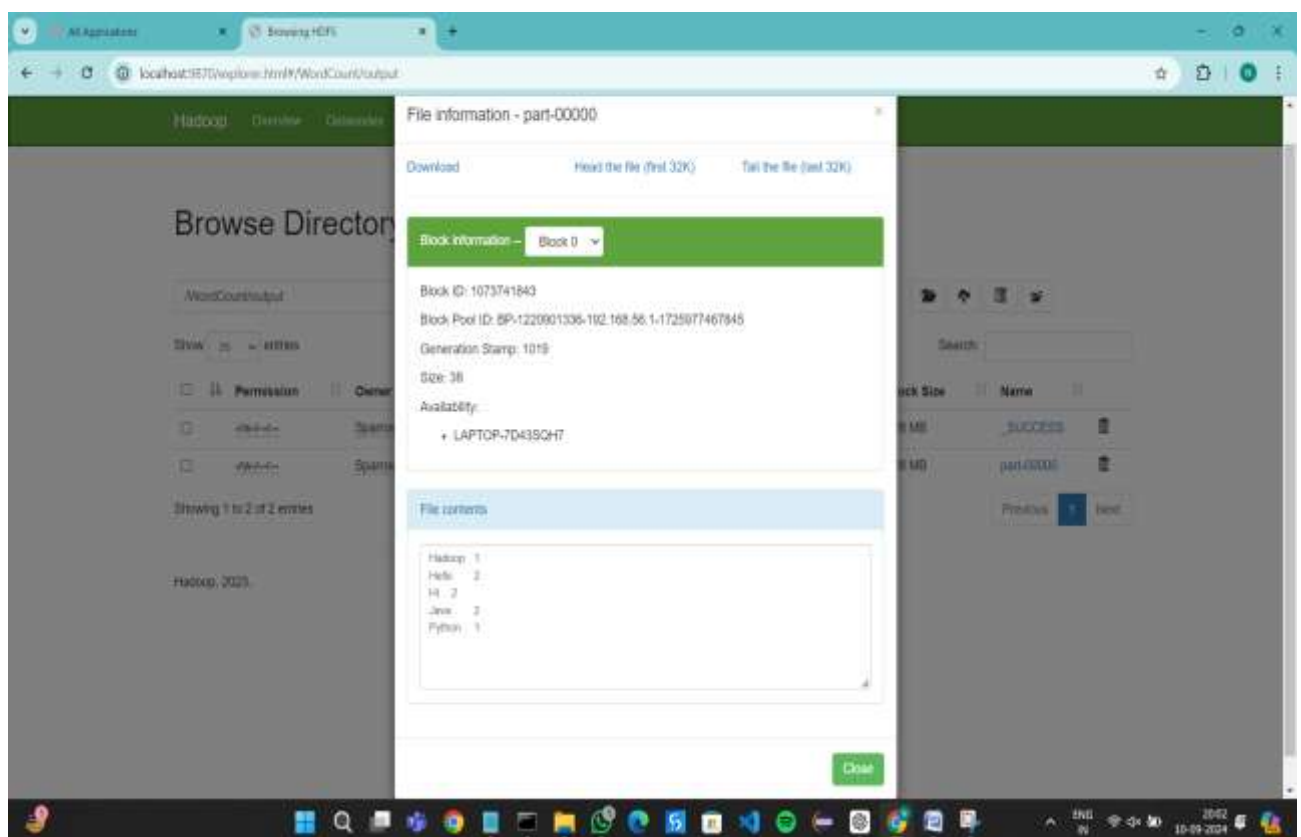
Administrator: Windows PowerShell
PS C:\Users\user> .\WordCount\bin\WordCount.exe

MapReduce Framework
Map input records=1
Map output records=1
Map output bytes=128
Map output materialized bytes=128
Input split bytes=128
Combine input records=0
Combine output records=0
Reduce input groups=0
Reduce shuffle bytes=0
Reduce input records=0
Reduce output records=0
Spilled Records=2
Failed Shuffles=0
Merge Map outputs=0
GC time elapsed (sec)=0.07
CPU time spent (sec)=0.08
Reduced memory (bytes)=4096000
Virtual memory (bytes) snapshot=147094400
Total committed heap usage (bytes)=103779840
Peak Map physical memory (bytes)=44288000
Peak Map virtual memory (bytes)=88588160
Peak Reduce physical memory (bytes)=20428800
Peak Reduce virtual memory (bytes)=44737600

Shuffle Summary
GID_10=0
Connections=0
IO_MBytes=0
WMB_MBytes=0
WMB_MBytes=0
WMB_MBytes=0
File Input Format Counters
Bytes Read=0
File Output Format Counters
Bytes Written=0

2024-09-09 10:57:57,588 INFO org.apache.hadoop.mapreduce.lib.output.FileOutputFormat: Output directory: /usr/local/hadoop/output
C:\Users\user>

```



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