

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.

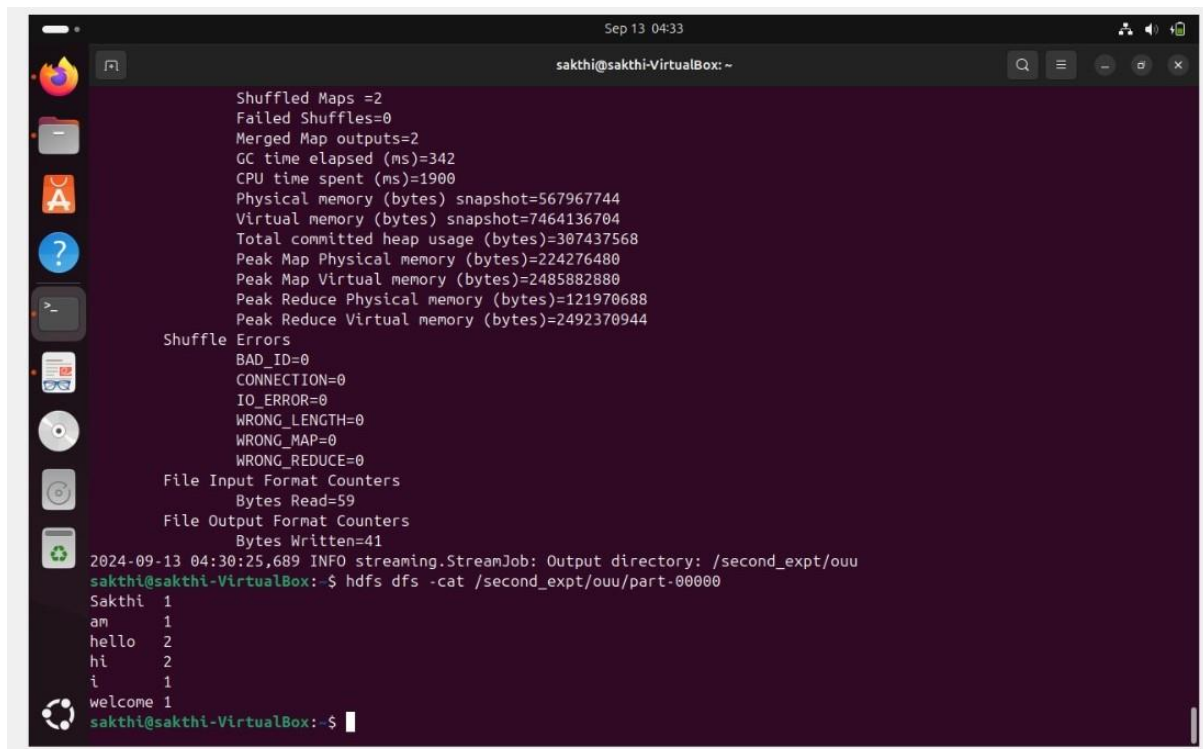
```
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \  
-input /word_count_in_python/word_count_data.txt \  
-output /word_count_in_python/new_output \  
-mapper /path/to/mapper.py \  
-reducer /path/to/reducer.py
```

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
```

OUTPUT:



```
Sep 13 04:33  
sakthi@sakthi-VirtualBox: ~  
  
Shuffled Maps =2  
Failed Shuffles=0  
Merged Map outputs=2  
GC time elapsed (ms)=342  
CPU time spent (ms)=1900  
Physical memory (bytes) snapshot=567967744  
Virtual memory (bytes) snapshot=7464136704  
Total committed heap usage (bytes)=307437568  
Peak Map Physical memory (bytes)=224276480  
Peak Map Virtual memory (bytes)=2485882880  
Peak Reduce Physical memory (bytes)=121970688  
Peak Reduce Virtual memory (bytes)=2492370944  
  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
  
File Input Format Counters  
Bytes Read=59  
File Output Format Counters  
Bytes Written=41  
2024-09-13 04:30:25,689 INFO streaming.StreamJob: Output directory: /second_expt/ouu  
sakthi@sakthi-VirtualBox: $ hdfs dfs -cat /second_expt/ouu/part-00000  
Sakthi 1  
am 1  
hello 2  
hi 2  
i 1  
welcome 1  
sakthi@sakthi-VirtualBox: $
```

Result:

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