

EXP 3: Map Reduce program to process a weather dataset.**AIM:**

To implement MapReduce program to process a weather dataset.

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.

Download the dataset (weather data)

Output:

```
hadoop@vishva-a-VirtualBox: ~
hadoop@vishva-a-virtualbox:~$ nano weather_data.txt
hadoop@vishva-a-VirtualBox:~$ cat weather_data.txt
Date,Location,Mintemp,MaxTemp,Rainfall,Sunshine,WindGustSpeed
2022-01-01,Loc1,8.0,24.3,0.0,6.3,30
2022-01-02,Loc2,14.0,26.9,3.6,9.7,39
2022-01-03,Loc3,13.7,23.4,3.6,3.3,85
2022-01-04,Loc4,13.3,15.5,39.8,9.1,54
2022-01-05,Loc5,7.6,16.1,2.8,10.6,50
2022-01-06,Loc6,6.2,16.9,0.0,8.2,44
2022-01-07,Loc7,6.1,18.2,0.2,8.4,43
2022-01-08,Loc8,8.3,17.0,0.0,4.6,41
2022-01-09,Loc9,8.8,19.5,0.0,4.1,48
2022-01-10,Loc10,8.4,22.8,16.2,7.7,31
2022-01-11,Loc11,9.1,25.2,0.0,11.9,30
2022-01-12,Loc12,8.5,27.3,0.2,12.5,41
2022-01-13,Loc13,10.1,27.9,0.0,13.0,30
2022-01-14,Loc14,12.1,30.9,0.0,12.4,44
2022-01-15,Loc15,10.1,31.2,0.0,13.1,41
2022-01-16,Loc16,12.4,32.1,0.0,11.1,46
2022-01-17,Loc17,13.8,31.2,0.0,8.4,44
2022-01-18,Loc18,11.7,30.0,1.2,10.1,52
2022-01-19,Loc19,12.4,32.2,0.0,12.0,30
```

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. split the line into words words = line.split()

```
#See the README hosted on the weather website which help us understand how each
position represents a column month = line[10:12] daily_max = line[38:45]
daily_max = daily_max.strip() # increase counters for word in words:
# write the results to STDOUT (standard output);
# what we output here will be go through the shuffle proess and then
# be the input for the Reduce step, i.e. the input for reducer.py
#
```

```

nano mapper.py
# Copy and paste the mapper.py code

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
# the mapper will get daily max temperature and group it by month. so output will be
(month,daily_max_temperature)

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # tab-delimited; month and daily max temperature as output
    print
('%s\t%s' % (month ,daily_max))

```

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/env python

from operator import itemgetter
import sys

#reducer will get the input from stdid which will be a collection of key, value(Key=month ,
value= daily max temperature)
#reducer logic: will get all the daily max temperature for a month and find max temperature
for the month
#shuffle will ensure that key are sorted(month)
current_month = None
current_max = 0
month = None

# input comes from STDIN for line
in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    month,
    daily_max = line.split('\t', 1)

```

```

    # convert daily_max (currently a string) to float    try:
        daily_max = float(daily_max)
except ValueError:
    # daily_max was not a number, so silently
    # ignore/discard this line        continue

    # this IF-switch only works because Hadoop shuffle process sorts map output
    # by key (here: month) before it is passed to the reducer
if current_month == month:        if daily_max >
current_max:        current_max = daily_max    else:        if
current_month:
    # write result to STDOUT        print
('%s\t%s' % (current_month, current_max))
current_max = daily_max
current_month = month

# output of the last month if current_month == month:
print ('%s\t%s' % (current_month, current_max))

```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

```
start-all.sh
```

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 the program using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the program using Hadoop Streaming.

```
hadoop fs -mkdir -p /weatherdata
```

```
hadoop fs -copyFromLocal /home/sx/Downloads/dataset.txt /weatherdata
```

```
hdfs dfs -ls /weatherdata
```

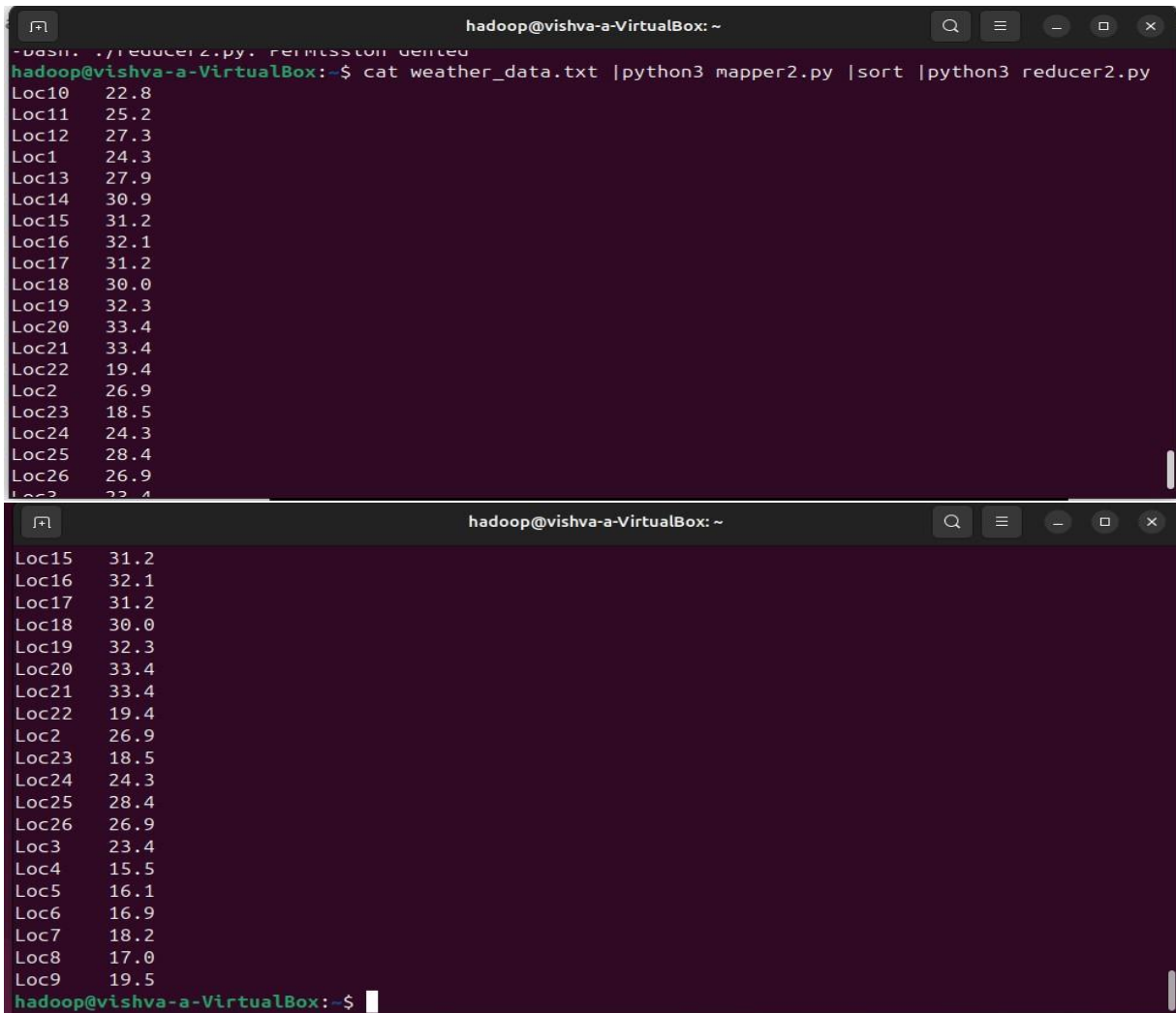
```
hadoop jar /home/sx/hadoop-3.2.3/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar \  
-input /weatherdata/dataset.txt \  
-output /weatherdata/output \  
-file "/home/sx/Downloads/mapper.py" \  
mapper "python3 mapper.py" \  
-file "/home/sx/Downloads/reducer.py" \  
-reducer "python3 reducer.py"
```

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/outputfile.txt
```

Step 8: Check Output:

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

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/output/ /part-00000
```



```
hadoop@vishva-a-VirtualBox: ~  
-bash: ./reducer2.py: Permission denied  
hadoop@vishva-a-VirtualBox:~$ cat weather_data.txt | python3 mapper2.py | sort | python3 reducer2.py  
Loc10 22.8  
Loc11 25.2  
Loc12 27.3  
Loc1 24.3  
Loc13 27.9  
Loc14 30.9  
Loc15 31.2  
Loc16 32.1  
Loc17 31.2  
Loc18 30.0  
Loc19 32.3  
Loc20 33.4  
Loc21 33.4  
Loc22 19.4  
Loc2 26.9  
Loc23 18.5  
Loc24 24.3  
Loc25 28.4  
Loc26 26.9  
Loc3 23.4  
Loc4 15.5  
Loc5 16.1  
Loc6 16.9  
Loc7 18.2  
Loc8 17.0  
Loc9 19.5  
hadoop@vishva-a-VirtualBox:~$
```

After copy and paste the above output in your local file give the below command to remove the directory from hdfs : `hadoop fs -rm -r /weatherdata/output`

Result:

Thus, the program for weather dataset using Map Reduce has been executed successfully.