**Exp.No.: 3 Map Reduce program to process a weather dataset**

**AIM:**

To implement MapReduce program to process a weather dataset.

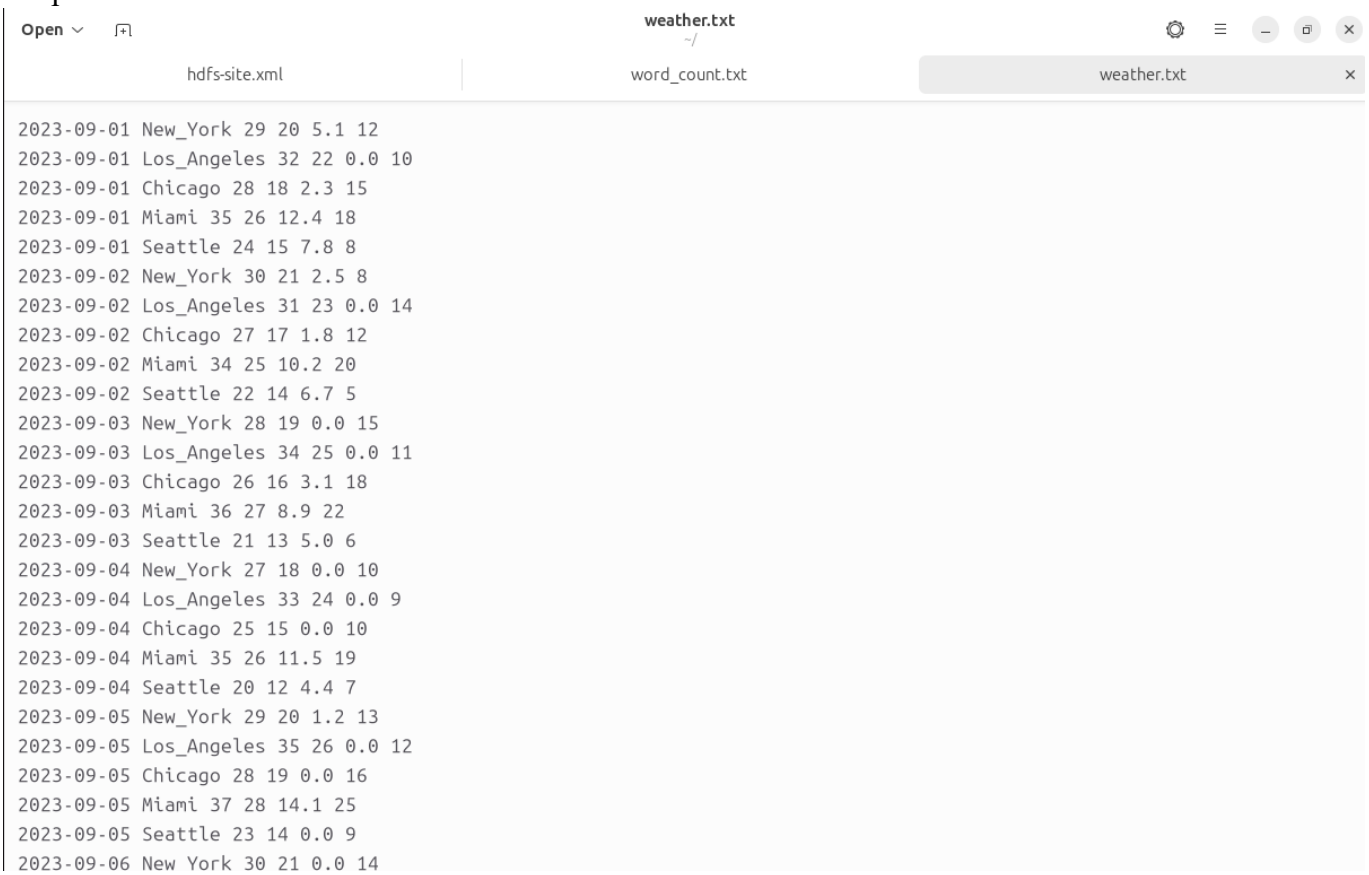
**Procedure:**

**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:




**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 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,dailymax_temperature)

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()     # 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
        #
        # 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
```

```
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/ha
doop-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 outpu
t 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_000000_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 outpu
t 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 program in the specified HDFS output directory.

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


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.