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:

Open weather_data.txt ~/weather

sample_weather.txt | weather_data.txt | mapper.py | hive-env.sh.template | hadoop-env.sh

```
2024-01-01  25.6
2024-01-02  26.1
2024-01-03  24.8
2024-01-04  22.7
2024-01-05  23.9
2024-02-01  28.5
2024-02-02  27.9
2024-02-03  26.7
2024-02-04  29.1
2024-03-01  31.2
2024-03-02  32.8
2024-03-03  30.4
2024-03-04  33.6
2024-04-01  34.5
2024-04-02  35.2
2024-04-03  33.9
2024-04-04  36.1
2024-05-01  40.0
2024-05-02  39.5
2024-05-03  41.2
2024-05-04  42.1
2024-06-01  43.6
```

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:

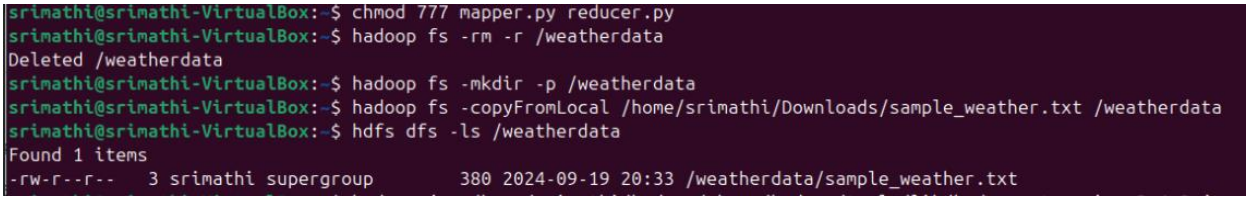
Start the Hadoop daemons and create a directory in HDFS to store your data.

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
```

```
srimathi@srimathi-VirtualBox: ~
srimathi@srimathi-VirtualBox:~$ hadoop jar /home/srimathi/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -input /weatherdata/sample_weather.txt
 -output /weatherdata/output -file /home/srimathi/mapper.py -mapper "python3 mapper.py" -file /home/srimathi/reducer.py -reducer "python3 reducer.py"
2024-09-19 20:33:51,174 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/srimathi/mapper.py, /home/srimathi/reducer.py] [] /tmp/streamjob4008441393485500949.jar tmpDir=null
2024-09-19 20:33:52,298 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-19 20:33:52,408 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-19 20:33:52,408 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-19 20:33:52,444 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-19 20:33:52,798 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-19 20:33:52,903 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-19 20:33:53,104 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2021541506_0001
2024-09-19 20:33:53,104 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-19 20:33:53,386 INFO mapred.LocalDistributedCacheManager: Localized file:/home/srimathi/mapper.py as file:/tmp/hadoop-srimathi/mapred/local/job_l
ocal2021541506_0001_ac3c7a01-5a5e-4f16-ae9a-717acdb0bce9/mapper.py
2024-09-19 20:33:53,419 INFO mapred.LocalDistributedCacheManager: Localized file:/home/srimathi/reducer.py as file:/tmp/hadoop-srimathi/mapred/local/job_
local2021541506_0001_445b85db-4411-4884-9f16-4cf1e3534c8e/reducer.py
2024-09-19 20:33:53,494 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-19 20:33:53,499 INFO mapreduce.Job: Running job: job_local2021541506_0001
2024-09-19 20:33:53,500 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-19 20:33:53,512 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-09-19 20:33:53,527 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-19 20:33:53,528 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup
 failures: false
2024-09-19 20:33:53,613 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-09-19 20:33:53,616 INFO mapred.LocalJobRunner: Starting task: attempt_local2021541506_0001_m_000000_0
2024-09-19 20:33:53,663 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-19 20:33:53,664 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup
 failures: false
2024-09-19 20:33:53,691 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-09-19 20:33:53,703 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/weatherdata/sample_weather.txt:0+380
2024-09-19 20:33:53,746 INFO mapred.MapTask: numReduceTasks: 1
2024-09-19 20:33:53,829 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
```

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

```
                Map output records=6
                Map output bytes=48
                Map output materialized bytes=66
                Input split bytes=104
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=66
                Reduce input records=6
                Reduce output records=2
                Spilled Records=12
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=11
                Total committed heap usage (bytes)=465567744
        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=515
        File Output Format Counters
                Bytes Written=16
2024-09-19 20:46:15,400 INFO streaming.StreamJob: Output directory: /weatherdata/output
srimathi@srimathi-VirtualBox:~$ hadoop fs -cat /weatherdata/output/part-00000
01      49.0
02      42.0
srimathi@srimathi-VirtualBox:~$ S
```

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.

Reg No:210701259