

**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:**

The screenshot shows a code editor window with the title 'weather\_data.txt' and a subtitle '~/.weather'. The editor contains a list of dates and temperatures, with the 'weather\_data.txt' tab selected. The data is as follows:

| Date       | Temperature |
|------------|-------------|
| 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,daily_max_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 process 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
```

```
hadoop@priyav-VirtualBox:~/weather$ chmod 777 mapper.py reducer.py
hadoop@priyav-VirtualBox:~/weather$ hadoop fs -mkdir -p /weatherdata
hadoop fs -copyFromLocal /home/hadoop/weather/weather_data.txt /weatherdata

hadoop@priyav-VirtualBox:~/weather$ hdfs dfs -ls /weatherdata
Found 1 items
-rw-r--r-- 3 hadoop supergroup 425 2024-09-02 11:13 /weatherdata/weather_data.txt
```

### 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@priyav-VirtualBox:~/weather$ hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -input /weatherdata/weather_data.txt -output /weatherdata/output -file /home/hadoop/weather/mapper.py -mapper "python3 mapper.py" -file /home/hadoop/weather/reducer.py -reducer "python3 reducer.py"
2024-09-02 11:38:53,541 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/hadoop/weather/mapper.py, /home/hadoop/weather/reducer.py] [] /tmp/streamjob5626630794616344731.jar
tmpDir=null
2024-09-02 11:38:54,341 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-09-02 11:38:54,570 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-09-02 11:38:54,570 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-09-02 11:38:54,583 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-09-02 11:38:54,934 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-02 11:38:55,026 INFO mapreduce.JobSubmitter: number of splits:1
2024-09-02 11:38:55,140 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local168524243_0001
2024-09-02 11:38:55,140 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-02 11:38:55,315 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hadoop/weather/mapper.py as file:/tmp/hadoop-hadoop/mapred/local/job_local168524243_0001_641a0cd5-5612-4dca-aa9a-448703987c4e/mapper.py
2024-09-02 11:38:55,361 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hadoop/weather/reducer.py as file:/tmp/hadoop-hadoop/mapred/local/job_local168524243_0001_1b146316-830c-47ac-b6a9-c4ed5198f393/reducer.py
2024-09-02 11:38:55,453 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-09-02 11:38:55,454 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-09-02 11:38:55,455 INFO mapreduce.Job: Running job: job_local168524243_0001
2024-09-02 11:38:55,463 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2024-09-02 11:38:55,470 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-09-02 11:38:55,471 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
```

### 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@priyav-VirtualBox:~/weather$ hadoop fs -ls /weatherdata/output
Found 2 items
-rw-r--r--  3 hadoop supergroup          0 2024-09-02 11:38 /weatherdata/output/_SUCCESS
-rw-r--r--  3 hadoop supergroup        48 2024-09-02 11:38 /weatherdata/output/part-00000
hadoop@priyav-VirtualBox:~/weather$ hadoop fs -cat /weatherdata/output/part-00000
01      26.1
02      29.1
03      33.6
04      36.1
05      42.1
06      44.0
hadoop@priyav-VirtualBox:~/weather$
```

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.