

Exp.No : 10

Date : 02/09/2024

IMPLEMENT THE MAX TEMPERATURE MAPREDUCE PROGRAM TO IDENTIFY THE YEAR WISE MAXIMUM TEMPERATURE FROM SENSOR

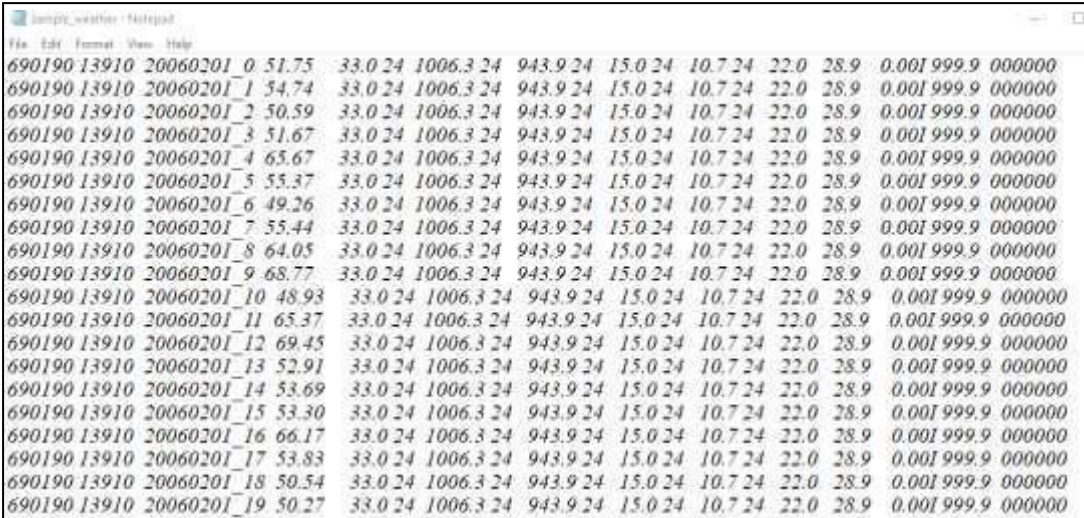
AIM:

To implement the max temperature Mapreduce program to identify the year wise maximum temperature from sensor.

PROCEDURE:

Step 1: Create Data File:

Create a file named "sample_weather.txt" and populate it with text data that you wish to analyse.



| | | | | | | | | | | | | | | | | | | |
|--------|-------|-------------|-------|------|----|--------|----|-------|----|------|----|------|----|------|------|-------|-------|--------|
| 690190 | 13910 | 20060201_0 | 51.75 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_1 | 54.74 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_2 | 50.59 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_3 | 51.67 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_4 | 65.67 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_5 | 55.37 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_6 | 49.26 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_7 | 55.44 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_8 | 64.05 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_9 | 68.77 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_10 | 48.93 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_11 | 65.37 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_12 | 69.45 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_13 | 52.91 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_14 | 53.69 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_15 | 53.30 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_16 | 66.17 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_17 | 53.83 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_18 | 50.54 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |
| 690190 | 13910 | 20060201_19 | 50.27 | 33.0 | 24 | 1006.3 | 24 | 943.9 | 24 | 15.0 | 24 | 10.7 | 24 | 22.0 | 28.9 | 0.001 | 999.9 | 000000 |

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.

mapper.py:

```
#!/usr/bin/python
3 import sys
def map1():
    for line in sys.stdin:
        tokens = line.strip().split()
        if len(tokens) < 13:
            continue
        station = tokens[0]
        if "STN" in station:
            continue
        date_hour = tokens[2]
        temp = tokens[3]
        dew = tokens[4]
        wind = tokens[12]
        if temp == "9999.9" or dew == "9999.9" or wind ==
```

```

"999.9":      continue      hour      =
int(date_hour.split("_")[-1])      date      =
date_hour[:date_hour.rfind("_")-2] if 4 <
hour <= 10: section = "section1"
elif 10 < hour <= 16:
    section = "section2"
elif 16 < hour <= 22:
    section = "section3"
else:
    section = "section4"
key_out = f"{station}_{date}_{section}"
value_out = f"{temp} {dew} {wind}"
print(f"{key_out}\t{value_out}")
if __name__ == "__main__":
    map1()

```

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.

reducer.py:

```

#!/usr/bin/python
3 import sys
def reduce1():
    current_key = None
    sum_temp, sum_dew, sum_wind = 0, 0, 0
    count = 0
    for line in sys.stdin:
        key, value = line.strip().split("\t")
        temp, dew, wind = map(float, value.split())
        if current_key is None:
            current_key = key
            if key == current_key:
                sum_temp += temp
                sum_dew += dew
                sum_wind += wind
                count += 1
            else:
                avg_temp = sum_temp / count
                avg_dew = sum_dew / count
                avg_wind = sum_wind / count
                print(f"{current_key}\t{avg_temp} {avg_dew} {avg_wind}")

                current_key = key
                sum_temp, sum_dew, sum_wind = temp, dew, wind
                count = 1
        if current_key is not None:
            avg_temp = sum_temp / count
            avg_dew = sum_dew / count
            avg_wind = sum_wind / count
            print(f"{current_key}\t{avg_temp} {avg_dew} {avg_wind}")
    if __name__ == "__main__":
        reduce1()

```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data. Run the following commands to store the data in the WeatherData Directory.

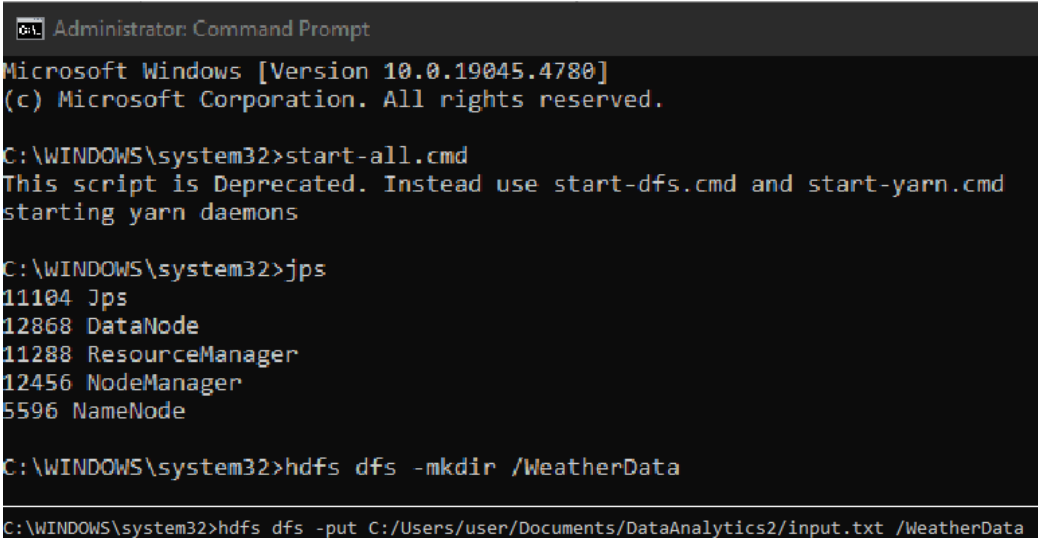
```
start-all.cmd cd C:/Hadoop/sbin hdfs dfs -mkdir /WeatherData hdfs dfs -put
C:/Users/user/Documents/DataAnalytics2/input.txt /WeatherData hadoop jar
C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar ^
-input /user/input/sample_weather.txt ^
-output /user/output ^
-mapper "python C:/Users/user/Documents/DataAnalytics2/mapper.py" ^ -reducer
"python C:/Users/user/Documents/DataAnalytics2/reducer.py"
```

Step 5: Check Output:

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

```
hdfs dfs -cat /WeatherData/output/part-00000
```

OUTPUT:



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\WINDOWS\system32>jps
11104 Jps
12868 DataNode
11288 ResourceManager
12456 NodeManager
5596 NameNode

C:\WINDOWS\system32>hdfs dfs -mkdir /WeatherData

C:\WINDOWS\system32>hdfs dfs -put C:/Users/user/Documents/DataAnalytics2/input.txt /WeatherData
```

