

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.

File	Edit	View
23907	20150101	2.423 -98.08 30.62 2.2 -0.6 0.8 0.9 7.0 1.47 C 3.7 1.1 2.5 99.9 85.4
97.2	0.369	0.308 -99.000 -99.000 -99.000 7.0 8.1 -9999.0 -9999.0 -9999.0
23907	20150102	2.423 -98.08 30.62 3.5 1.3 2.4 2.2 10.2 1.43 C 4.9 2.3 3.1 100.0 98.8
99.8	0.391	0.327 -99.000 -99.000 -99.000 7.1 7.9 -9999.0 -9999.0 -9999.0
23907	20150103	2.423 -98.08 30.62 15.9 2.3 9.1 7.5 3.1 11.00 C 16.4 2.9 7.3 100.0 34.8
73.7	0.450	0.397 -99.000 -99.000 -99.000 7.6 7.9 -9999.0 -9999.0 -9999.0
23907	20150104	2.423 -98.08 30.62 9.2 -1.3 3.9 4.2 0.0 13.24 C 12.4 -0.5 4.9 82.0 40.6
61.7	0.413	0.352 -99.000 -99.000 -99.000 7.3 7.9 -9999.0 -9999.0 -9999.0
23907	20150105	2.423 -98.08 30.62 10.9 -3.7 3.6 2.6 0.0 13.37 C 14.7 -3.0 3.8 77.9 33.3
57.4	0.399	0.340 -99.000 -99.000 -99.000 6.3 7.0 -9999.0 -9999.0 -9999.0
23907	20150106	2.423 -98.08 30.62 20.2 2.9 11.6 10.9 0.0 12.90 C 22.0 1.6 9.9 67.7 30.2
49.3	0.395	0.335 -99.000 -99.000 -99.000 8.0 8.0 -9999.0 -9999.0 -9999.0
23907	20150107	2.423 -98.08 30.62 10.9 -3.4 3.8 4.5 0.0 12.68 C 12.4 -2.1 5.5 82.7 36.5
55.7	0.387	0.328 -99.000 -99.000 -99.000 7.6 8.3 -9999.0 -9999.0 -9999.0
23907	20150108	2.423 -98.08 30.62 0.6 -7.9 -3.6 -3.3 0.0 4.98 C 3.9 -4.8 -0.5 57.7 37.6
48.1	0.372	0.316 -99.000 -99.000 -99.000 4.7 6.1 -9999.0 -9999.0 -9999.0
23907	20150109	2.423 -98.08 30.62 2.0 0.1 1.0 0.8 0.0 2.52 C 4.1 1.2 2.5 87.8 48.9
64.4	0.368	0.312 -99.000 -99.000 -99.000 5.4 6.2 -9999.0 -9999.0 -9999.0
23907	20150110	2.423 -98.08 30.62 0.5 -2.0 -0.8 -0.6 3.9 2.11 C 2.5 -0.1 1.4 99.9 47.7
85.8	0.373	0.314 -99.000 -99.000 -99.000 5.1 6.0 -9999.0 -9999.0 -9999.0
23907	20150111	2.423 -98.08 30.62 10.9 0.0 5.4 4.4 2.6 6.38 C 12.7 1.3 5.8 100.0 77.8
97.1	0.420	0.362 -99.000 -99.000 -99.000 6.5 6.7 -9999.0 -9999.0 -9999.0
23907	20150112	2.423 -98.08 30.62 6.5 1.4 4.0 4.3 0.0 1.55 C 6.9 2.7 5.1 100.0 89.4
97.8	0.412	0.350 -99.000 -99.000 -99.000 7.3 7.5 -9999.0 -9999.0 -9999.0
23907	20150113	2.423 -98.08 30.62 3.0 -0.7 1.1 1.2 0.0 3.26 C 5.6 0.7 2.9 99.7 80.7
90.7	0.401	0.337 -99.000 -99.000 -99.000 6.1 6.8 -9999.0 -9999.0 -9999.0
23907	20150114	2.423 -98.08 30.62 2.9 0.9 1.9 1.8 0.7 1.88 C 4.7 2.0 3.1 99.6 90.8
97.9	0.395	0.331 -99.000 -99.000 -99.000 6.1 6.7 -9999.0 -9999.0 -9999.0
23907	20150115	2.423 -98.08 30.62 13.2 1.2 7.2 6.4 0.0 13.37 C 16.4 1.4 6.7 98.9 46.7
73.4	0.395	0.333 -99.000 -99.000 -99.000 6.7 7.0 -9999.0 -9999.0 -9999.0
23907	20150116	2.423 -98.08 30.62 16.7 3.5 10.1 9.9 0.0 13.68 C 19.2 1.3 8.7 80.2 38.1
58.2	0.391	0.330 -99.000 -99.000 -99.000 7.3 7.4 -9999.0 -9999.0 -9999.0
23907	20150117	2.423 -98.08 30.62 19.5 5.0 12.2 12.3 0.0 10.96 C 20.9 3.3 10.6 87.7 30.4
55.7	0.388	0.327 -99.000 -99.000 -99.000 8.7 8.4 -9999.0 -9999.0 -9999.0
23907	20150118	2.423 -98.08 30.62 20.9 7.6 14.3 13.7 0.0 15.03 C 23.4 3.5 11.9 45.9 14.6
31.4	0.383	0.325 -99.000 -99.000 -99.000 9.5 9.2 -9999.0 -9999.0 -9999.0

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)

Download the dataset (weather data)

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 5: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

```
chmod 777 mapper.py reducer.py
```

Step 6: 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" \
```

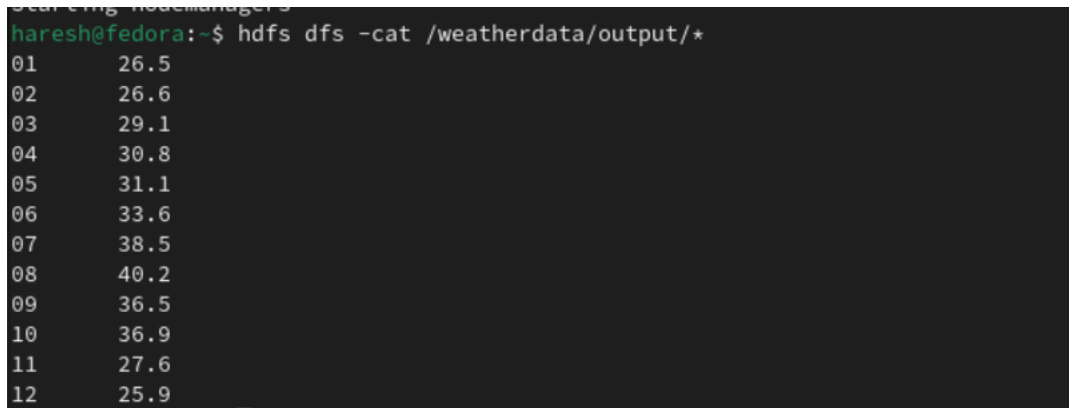
```
-reducer "python3 reducer.py"
```

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

Step 7: 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
```

A terminal window screenshot showing the command 'hdfs dfs -cat /weatherdata/output/*' being executed. The output is a list of 12 rows, each with an index from 01 to 12 and a corresponding temperature value. The values are: 26.5, 26.6, 29.1, 30.8, 31.1, 33.6, 38.5, 40.2, 36.5, 36.9, 27.6, and 25.9.

01	26.5
02	26.6
03	29.1
04	30.8
05	31.1
06	33.6
07	38.5
08	40.2
09	36.5
10	36.9
11	27.6
12	25.9

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.