

EXP 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:

*sample_weather.txt ~/Downloads										
1	5690190	13910	20060201_0	51.75	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
2	690190	13910	20060201_1	54.74	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
3	690190	13910	20060201_2	50.59	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
4	690190	13910	20060201_3	51.67	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
5	690190	13910	20060201_4	65.67	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
6	690190	13910	20060201_5	55.37	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
7	690190	13910	20060201_6	49.26	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
8	690190	13910	20060201_7	55.44	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
9	690190	13910	20060201_8	64.05	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
10	690190	13910	20060201_9	68.77	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
11	690190	13910	20060201_10	48.93	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
12	690190	13910	20060201_11	65.37	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
13	690190	13910	20060201_12	69.45	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
14	690190	13910	20060201_13	52.91	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
15	690190	13910	20060201_14	53.69	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
16	690190	13910	20060201_15	53.30	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
17	690190	13910	20060201_16	66.17	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
18	690190	13910	20060201_17	53.83	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
19	690190	13910	20060201_18	50.54	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
20	690190	13910	20060201_19	50.27	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
21	690190	13910	20060201_20	59.08	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
22	690190	13910	20060201_21	53.05	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24
23	690190	13910	20060201_22	57.97	33.0	24	1006.3	24	943.9	24
	28.9	0.00I	999.9	000000					15.0	24

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 = 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
```

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
```

Step 8: Check Output:

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

Block information -- Block 0 ▾

Block ID: 1073741852

Block Pool ID: BP-964807653-127.0.1.1-1724072259661

Generation Stamp: 1028

Size: 303

Availability:

- ubuntu.myquest.virtualbox.org

File contents

690190_200602_section1	53.87166666666665	25.900000000000006
7.774999999999995		
690190_200602_section2	54.76125	25.900000000000006
7.774999999999998		
690190_200602_section3	53.25041666666666	25.900000000000006
7.774999999999998		
690190_200602_section4	52.44708333333333	25.900000000000002
7.774999999999998		

Result:

Thus, the program for weather dataset using Map Reduce has been executed successfully.