

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:



```
1 20210101 25 0
2 20210102 30 5
3 20210201 28 0
4 20210202 32 2
5 20210301 33 3
```

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 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))
```

```
varsh@ubuntu: ~$ ./start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as varsh in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
Starting resourcemanager
Starting nodemanager
varsh@ubuntu: ~$ jps
4803 NameNode
5444 NodeManager
5222 ResourceManager
5118 SecondaryNameNode
5838 Jps
4927 DataNode
varsh@ubuntu: ~$ nano weather.txt
varsh@ubuntu: ~$ nano mapper1.py
varsh@ubuntu: ~$ nano reducer1.py
varsh@ubuntu: ~$ chmod 777 mapper1.py reducer1.py
varsh@ubuntu: ~$ hdfs dfs -mkdir -p /weatherdata
varsh@ubuntu: ~$ hdfs dfs -copyfromLocal /home/varsh/weather.txt /weatherdata
varsh@ubuntu: ~$ hdfs dfs -ls /weatherdata
Found 1 items
varsh@ubuntu: ~$ cat /home/varsh/weather.txt | python3 /home/varsh/reducer.py
2021 25
2021 30
2021 28
2021 32
2021 33
varsh@ubuntu: ~$ hadoop jar /home/varsh/Downloads/jar_files/hadoop-streaming-2.7.3.jar \
-input /weatherdata/weather.txt \
-output /weatherdata/weather_output \
-mapper /home/varsh/mapper1.py \
-reducer /home/varsh/reducer1.py
packageJobJar: [/tmp/hadoop-unjar0317137089188243470/] [/tmp/streamjob556316645680774582.jar tmpDir=null]
2024-09-19 13:04:19,291 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-19 13:04:19,280 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-19 13:04:19,997 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/varsh/.staging/job_172673848866_0001
```

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

```

varsh@ubuntu:~$ hadoop jar /home/varsh/Downloads/jar_files/hadoop-streaming-2.7.3.jar \
-input /weatherdata/weather.txt \
-output /weatherdata/output \
-mapper "/usr/bin/python3 /home/varsh/mapper.py" \
-reducer "/usr/bin/python3 /home/varsh/reducer.py"
package.json: ["/tmp/hadoop-unjar451607168651095742/"] [] /tmp/streamjob5542878243039637519.jar tmpDir=null
2024-09-19 13:19:59,208 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-19 13:19:59,660 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-19 13:20:00,274 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/varsh/.stagi
ng/job_1726730486866_0002
2024-09-19 13:20:00,951 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-19 13:20:01,204 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-19 13:20:02,090 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1726730486866_0002
2024-09-19 13:20:02,090 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-19 13:20:02,672 INFO conf.Configuration: resource-types.xml not found
2024-09-19 13:20:02,673 INFO resource.ResourceUtils: unable to find 'resource-types.xml'
2024-09-19 13:20:02,850 INFO impl.VarnClientImpl: Submitted application application_1726730486866_0002
2024-09-19 13:20:02,952 INFO mapreduce.Job: The url to track the job: http://ubuntu.myguest.virtualbox.org:8088/proxy/application_17
26730486866_0002/
2024-09-19 13:20:02,969 INFO mapreduce.Job: Running job: job_1726730486866_0002
2024-09-19 13:20:22,014 INFO mapreduce.Job: Job job_1726730486866_0002 running in uber mode : false
2024-09-19 13:20:22,023 INFO mapreduce.Job: map 0% reduce 0%
2024-09-19 13:20:45,980 INFO mapreduce.Job: map 100% reduce 0%
2024-09-19 13:21:00,453 INFO mapreduce.Job: map 100% reduce 100%
2024-09-19 13:21:02,592 INFO mapreduce.Job: Job job_1726730486866_0002 completed successfully
2024-09-19 13:21:04,492 INFO mapreduce.Job: Counters: 54
File System Counters
FILE: Number of bytes read=56
FILE: Number of bytes written=717459
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=299
HDFS: Number of bytes written=8
HDFS: Number of read operations=11
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
HDFS: Number of bytes read erasure-coded=0
Job Counters
Launched map tasks=2

```

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" \
```

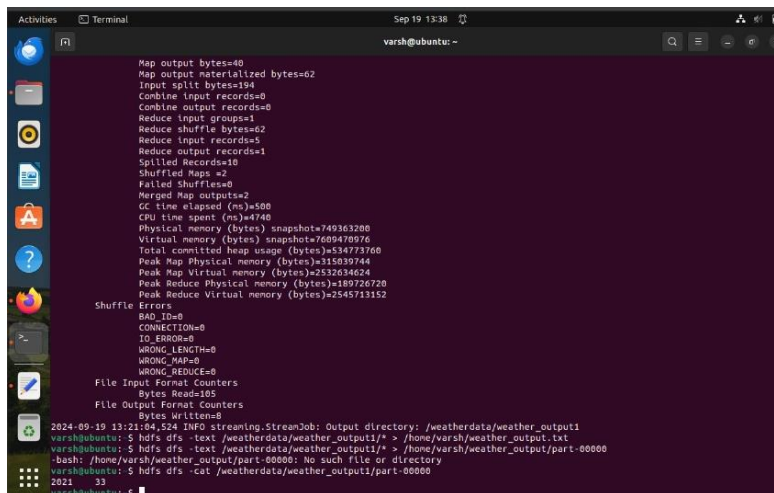
```
-reducer "python3 reducer.py"
```

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

Step 8: Check Output:

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/output/ /part-00000
```

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`



```
Map output bytes=40
Map output materialized bytes=62
Input split bytes=194
Combine input records=0
Combine output records=0
Reduce input groups=1
Reduce shuffle bytes=62
Reduce input records=5
Reduce output records=1
Spilled Records=10
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=500
CPU time spent (ms)=4748
Physical memory (bytes) snapshot=749363200
Virtual memory (bytes) snapshot=7609470976
Total committed heap usage (bytes)=4534773760
Peak Map Physical memory (bytes)=315839744
Peak Map Virtual memory (bytes)=2532634624
Peak Reduce Physical memory (bytes)=189726720
Peak Reduce Virtual memory (bytes)=2545713152

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=105
File Output Format Counters
Bytes Written=8

2024-09-19 13:21:04,524 INFO Streaming.StreamJob: Output directory: /weatherdata/weather_output1
varsh@ubuntu: $ hdfs dfs -text /weatherdata/weather_output1/* > /home/varsh/weather_output.txt
varsh@ubuntu: $ hdfs dfs -text /weatherdata/weather_output1/* > /home/varsh/weather_output/part-00000
-bash: /home/varsh/weather_output/part-00000: No such file or directory
varsh@ubuntu: $ hdfs dfs -cat /weatherdata/weather_output1/part-00000
2021 33
varsh@ubuntu: $
```

Result:

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