Ex No 3 **Map Reduce program to process a weather dataset.**

AIM:

To implement MapReduce program to process a weather dataset.

PROCEDURE:

1. Start Hadoop Services

Make sure you are in the `sbin` folder of Hadoop. Start the Hadoop services by running the following commands:

```
cd /usr/local/Cellar/hadoop/3.4.0/libexec/sbin
./start-dfs.sh ./start-yarn.sh
```

2. Prepare Your Files

Create the necessary files (`mapper.py`, `reducer.py`, and `sample_weather.txt`) on your local machine or the server where Hadoop is installed.

Create `sample_weather.txt`:

You can create this file in the current directory:

```
nano sample_weather.txt
```

Example data for `sample_weather.txt`:

```
STN001 2023-09-10_04 15.0 12.0 5.0
STN001 2023-09-10_10 25.0 20.0 8.0
STN002 2023-09-10_16 30.0 25.0 10.0
STN002 2023-09-10_22 22.0 18.0 7.0
```

Create `mapper.py`: `nano mapper.py`

Content for `mapper.py`:

```
#!/usr/bin/python3
import sys def map1():
```

```
    for line in sys.stdin: tokens = line.strip().split() if
        len(tokens) < 4: continue station = tokens[0] date_hour =
        tokens[1] temp = tokens[2] dew = tokens[3] wind =
        tokens[4] if len(tokens) > 4 else "999.9" 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("_")] 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()
```

Create

reducer.py: nano

reducer.py

Content for `reducer.py`:

```
#!/usr/bin/python3 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()
```

3. Upload Files to HDFS

Next, move your data file to HDFS so that it can be processed by the Hadoop MapReduce job.

Create HDFS Directory:

```
hdfs dfs -mkdir /WeatherData
```

Upload the Input Data (`sample_weather.txt`) to HDFS: `hdfs`

```
dfs -put sample_weather.txt /WeatherData
```

Verify the file upload:

```
hdfs dfs -ls /WeatherData
```

You should see something like:

```
Found 1 items
-rw-r--r-- 3 user group     1234 2024-09-11 12:00
/WeatherData/sample_weather.txt
```

4. Run the MapReduce Job

Now that your input file is in HDFS and your `mapper.py` and `reducer.py` are ready, you can run the MapReduce job.

Ensure you are still in the directory where your `mapper.py` and `reducer.py` scripts are located.

Run the Hadoop Streaming Job:

```
hadoop jar
/usr/local/Cellar/hadoop/3.4.0/libexec/share/hadoop/tools/lib/hadoop-s
treaming-3.4.0.jar \
-input /WeatherData/sample_weather.txt \
-output /WeatherData/output \ -mapper
"python3 mapper.py" \ reducer
"python3 reducer.py"
```

This command tells Hadoop to:

- Take the input from `/WeatherData/sample_weather.txt` on HDFS.
- Use `mapper.py` as the mapper script.
- Use `reducer.py` as the reducer script.
- Output the results to `/WeatherData/output`.

Note:

Ensure that both `mapper.py` and `reducer.py` have executable permissions. If not, make them executable by running:

```
chmod +x mapper.py reducer.py
```

5. View the Output

After the job completes, you can check the output that was stored in HDFS.

List the output directory:

```
hdfs dfs -ls /WeatherData/output
```

You should see something like:

```
Found 1 items
-rw-r--r--  3  user  group  456  2024-09-11  12:20
```

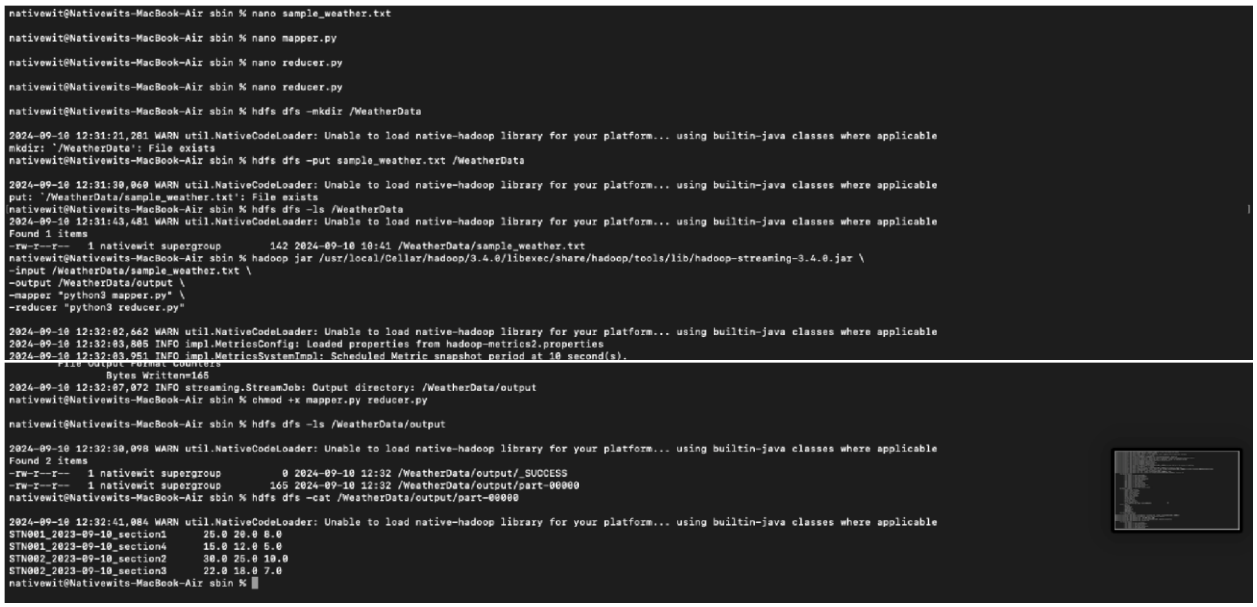
`/WeatherData/output/part-00000` **View the output data:**

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

This will print the final result of the MapReduce job. You should see output similar to:

```
STN001_2023-09-10_section1 15.0 12.0 5.0
STN001_2023-09-10_section2 25.0 20.0 8.0
STN002_2023-09-10_section3 30.0 25.0 10.0
STN002_2023-09-10_section4 22.0 18.0 7.0
```

OUTPUT:



RESULT:

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