Report

Pseudo Codes

1. **NoCombiner.java**

```
Class MinMaxTemp extends Writable
        Double minTemp, maxTemp;
        Public toString()
                Return minTemp and maxTemp;
        //It will override other Writable functions too


Mapper(key, value)
Create MinMaxTemp object as mmt
        String token[] = value.split(“,”);
        If token[2] = “TMAX”
                Set token[3] as tmax and -9999 as tmin
        Else if token[2] = “TMIN”
                Set token[3] as tmin and -9999 as tmax
        Set tmax and tmin to object of class MinMaxTemp
emit(stationId, MinMaxTemp)


Reducer(StationId, { MinMaxTemp1, MinMaxTemp2, …}) – (key,value)
Create MinMaxTemp object as final output value
        For all the objects in value
                Sum over tmax, tmin from object
                If tmax == -9999
```

Increment tmin

Else

Increment tmax

Update MinMaxTemp object with new values.

Emit(stationId, MinMaxTemp) // stationId, minTemp, maxTemp

2. **WithCombiner.java**

Class MinMaxTemp extends Writable

Double minTemp, maxTemp;

Integer minCount = 1, maxCount =1 ; // Initialized to 1, if combiner is not called

Public toString()

Return minTemp and maxTemp;

//It will override other Writable functions too

Mapper(key, value)

Create MinMaxTemp object as mmt

String token[] = value.split(“,”);

If token[2] = “TMAX”

Set token[3] as tmax and -9999 as tmin

Else if token[2] = “TMIN”

Set token[3] as tmin and -9999 as tmax

Set tmax and tmin to object of class MinMaxTemp

emit(stationId, MinMaxTemp)

```
Combiner(StationId, { MinMaxTemp1, MinMaxTemp2, …}) – (key,value)
        For all the objects in value
                Sum over tmax, tmin from object
                If tmax == -9999
                        Increment tmin
                Else
                        Increment tmax
        Emit(stationId, MinMaxTemp) //  stationId, minTemp, maxTemp
```

```
Reducer(StationId, { MinMaxTemp1, MinMaxTemp2, …}) – (key,value)
        Create MinMaxTemp object as final output value
        For all the objects in value
                Sum over tmax, tmin from object
                If obj.getMaxTemp == -9999
                        maxCount++;
                        maxCount += object.getMaxCount()
                if obj.getMinTemp == -9999
                        minCount++
                        minCount += object.getMinCount()
        Update MinMaxTemp object with new values
```

Emit(stationId, MinMaxTemp) // stationId, minTemp, maxTemp

3. **InMapperCombiner.java**

```
Class MinMaxTemp extends Writable
        Double minTemp, maxTemp;
        Public toString()
                Return minTemp and maxTemp;
        //It will override other Writable functions too

Mapper(key, value)
Create MinMaxTemp object as mmt
Create hashmap H
        String token[] = value.split(“,”);
If H.containsKey(stationId)
        Mmt = H.get(stationId)
        Initialize minSum, maxSum, minCount, maxCount from mmt

If token[2] = “TMAX”
                Set token[3] as tmax and -9999 as tmin
Else if token[2] = “TMIN”
                Set token[3] as tmin and -9999 as tmax
                Set tmax and tmin to object of class MinMaxTemp
emit(stationId, MinMaxTemp)
```

```
Reducer(StationId, { MinMaxTemp1, MinMaxTemp2, …}) – (key,value)
Create MinMaxTemp object as final output value
        For all the objects in value
                Sum over tmax, tmin from object
                If tmax == -9999
                        Increment tmin
                Else
                        Increment tmax
        Update MinMaxTemp object with new values.
Emit(stationId, MinMaxTemp) //  stationId, minTemp, maxTemp
```

For Program 2

- **Secondary Sort (Pseudo Code)**

```
Class MinMaxTemp extends Writable
        Double minTemp, maxTemp;
        Public toString()
                Return minTemp and maxTemp;
        //It will override other Writable functions too

// it has two variables for key – stationId and temperature
Class CompositKey extends Writable, WritableComparable<CompositeKey>
```

```
        String stationId, year;
        Override compareTo()
        Public int compareTo(CompositeKey k2)
                Result = Compare keys k1 and k2 on the basis of stationId
                If same , then
                        Result =  compare on the basis of year
        Return result;


Mapper(key, value)
Create MinMaxTemp object as mmt
        String token[] = value.split(“,”);
        Int year  = token[1].substring(0,4);
        If token[2] = “TMAX”
                Set token[3] as tmax and -9999 as tmin
        Else if token[2] = “TMIN”
                Set token[3] as tmin and -9999 as tmax
        Set tmax and tmin to object of class MinMaxTemp
emit(CompositeKey(stationId,year) ,  MinMaxTemp) // output of map


// partitioner will partition on basis of stationId only
// as we need to send same stationId to same reducer
Public int getPartitioner(CompositeKey, MinMaxTemp, numTasks)
        Return Math.abs(key.stationId.hashCode()%numTasks)
```

```
// Key sort Comparator
Public int compare(WritableComparable w1, WritableComparable w2)
        Result = compare stationId of w1 and w2
         If same,
                Result = compare year of w1 and w2
        Return result
// Grouping comparator
// we want to group on the basis of stationId only, as we want our records output
// as per grouped by stationId
Public int compare(WritableComparable w1, WritableComparable w2)
        Result = compare stationId of w1 and w2
        Return result


// reducer will get the data as per stationId and data will come as sorted by year
Reducer(StationId, {year1, tmin, tmax } , {year1, tmin,tmax} , {year2, tmin,tmax}
…) – (key,value)
Create MinMaxTemp object as final output value
Year  = key.getYear()
        For all the objects in value
// check for change in year
        If !(year = key.getYear())
                Update string for previous year data
                Update year = key.getYear();
                Initialize minSum, maxSum, minCount, maxCount to 0
```

If obj.getMaxTemp == -9999

maxCount++;

maxCount += object.getMaxCount()

if obj.getMinTemp == -9999

minCount++

minCount += object.getMinCount()

Update StringBuilder object with new values.

Emit(stationId, String) // (stationId, {1980,minTemp,maxTemp} , …. {1989, minTemp,maxTemp})

- Data will come to reducer for stationId , and related data would be sorted by year as sorting is done by both stationId and year while grouping is done by stationId only.
- So it will solve our purpose of Secondary Sort.

For example –

Data -> map – emit(ABC, 1980, 45 67)

EMIT(ABC,1980,52,85)

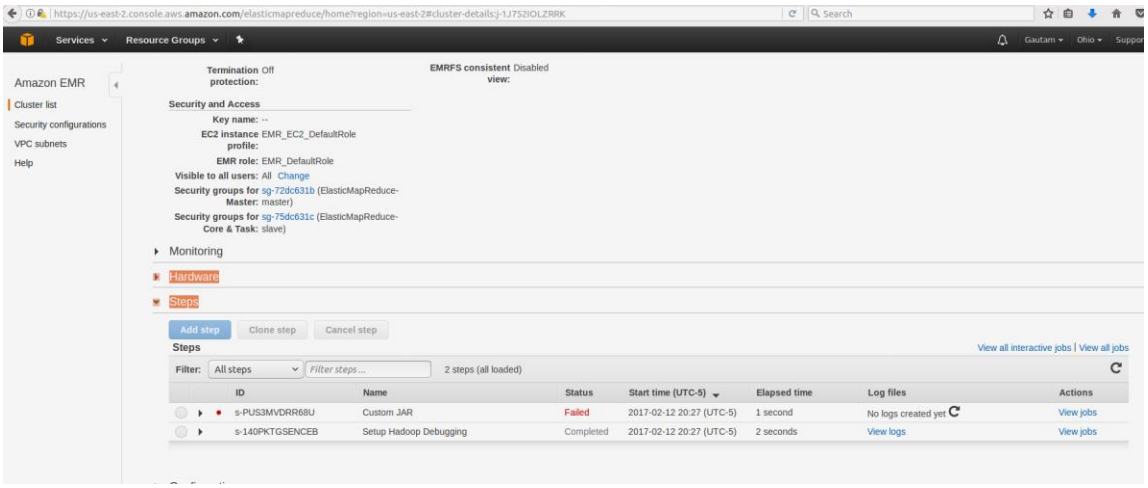
Emit(ABC, 1981, 78, 90)

Reducer(ABC, {1980, 45,67} , {1980, 52, 85}, {1981,78,90})

Emit(ABC, {1980, 48.5, 76}, {1981,78, 90})

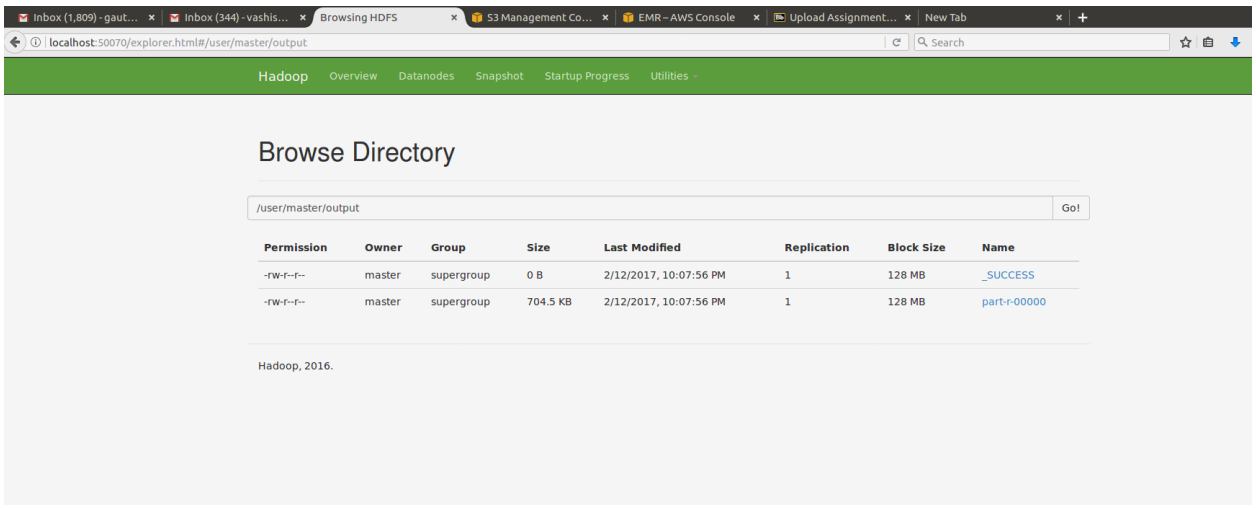
- **Performance Comparison**

I am not able to run my program on AWS, it is getting erred out without showing me any log. I am not able to debug that.



I have created the makefile and successfully ran all the programs on the local machine using alone and pseudo both.

Output Created on local host for NoCombiner using make pseudo command–



- So, I am not able to answer the performance questions using EMR run but just by executing programs on local machines –

1. In program Combiner, it will be called but there is no guarantee that it will always run. Combiner can't run more than once per map task as combiner may or may not be called after completion of map task only.

2. Combiner program will combine the data before sending it to reducer. Therefore, it will reduce the amount of data used in shuffle and sort phase and the data transferred to reducer from mapper.

3. In this case, local aggregation will be effective as transferred data between mapper and reducer is getting reduced. But for other cases it might be the issue if heap memory for hash map get overfilled.

4. InMapperCombiner will run in less time as it is combining data in mapper phase only and just transfering the single value for each station ID using HashMap. While Combiner program is using combiner to sum up the temperatures for station but without any guarantee that it will do the same for all the map outputs.

- I have added output files from my local machine run.