**Assignment 6 - Introduction to Map-Reduce (Hadoop)**

**Name: Shweta Pathak**

**UTA ID: 1001154572**

**Net Id: ssp4572**

**Section: 1.00 pm to 3.00 pm**

**References:**

1. <http://geetanjali-badhla.blogspot.com/2014/07/mapreduce-program-to-find-average.html>
2. <http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>
3. <https://www.youtube.com/watch?v=608lAGNUXs0>

**Code:**

1. **Code to calculate yearly average temperature.**

**public** **class** TemperatureMapper **extends** MapReduceBase **implements** Mapper<LongWritable, Text, Text, DoubleWritable>

{

**public** **static** **final** **int** ***MISSING*** = -9999;

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, DoubleWritable> output, Reporter reporter)

**throws** IOException

{

String line = value.toString();

String [] path = line.split(",");

String year = String.valueOf(Integer.parseInt(path[0]));

Double temperature;

Double Tmaxtemp = Double.*parseDouble*(path[4].toString());

Double Tmintemp = Double.*parseDouble*(path[5].toString());

temperature = (Tmaxtemp+Tmintemp)/2;

**if**(Tmaxtemp != ***MISSING*** && Tmintemp != ***MISSING***) {

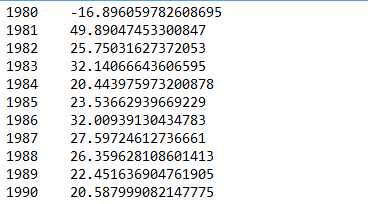
output.collect(**new** Text(year), **new** DoubleWritable(temperature));

}

}

}

**Output:**



**Time taken with 1 mapper 1 reducer:** 31098 milliseconds

1. **Code to calculate monthly average temperature year by year.**

**public** **class** TemperatureMapper **extends** MapReduceBase **implements** Mapper<LongWritable, Text, Text, DoubleWritable>

{

**public** **static** **final** **int** ***MISSING*** = -9999;

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, DoubleWritable> output, Reporter reporter)

**throws** IOException

{

String line = value.toString();

String [] path = line.split(",");

**int** month = Integer.*parseInt*(path[1]);

String year = String.*valueOf*(Integer.*parseInt*(path[0]));

**switch**(month)

{

**case** 1: year = year.concat("Jan");

**break**;

**case** 2: year = year.concat("Feb");

**break**;

**case** 3: year = year.concat("Mar");

**break**;

**case** 4: year = year.concat("April");

**break**;

**case** 5: year = year.concat("May");

**break**;

**case** 6: year = year.concat("Jun");

**break**;

**case** 7: year = year.concat("Jul");

**break**;

**case** 8: year = year.concat("Aug");

**break**;

**case** 9: year = year.concat("Sep");

**break**;

**case** 10: year = year.concat("Oct");

**break**;

**case** 11: year = year.concat("Nov");

**break**;

**case** 12: year = year.concat("Dec");

**break**;

**default**: **break**;

}

Double temperature;

Double Tmaxtemp = Double.*parseDouble*(path[4].toString());

Double Tmintemp = Double.*parseDouble*(path[5].toString());

temperature = (Tmaxtemp+Tmintemp)/2;

**if**(Tmaxtemp != ***MISSING*** && Tmintemp != ***MISSING***) {

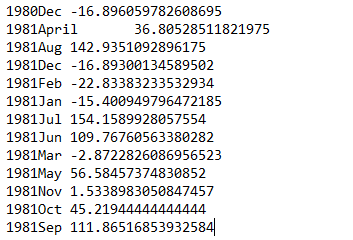
output.collect(**new** Text(year), **new** DoubleWritable(temperature));

}

}

}

**Output:**



**Time taken with 1 mapper 1 reducer:** 31017 milliseconds

1. **Code to calculate seasonal average temperature year by year.**

**public** **class** TemperatureMapper **extends** MapReduceBase **implements** Mapper<LongWritable, Text, Text, DoubleWritable>

{

**public** **static** **final** **int** ***MISSING*** = -9999;

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, DoubleWritable> output, Reporter reporter)

**throws** IOException

{

String line = value.toString();

String [] path = line.split(",");

**int** month = Integer.*parseInt*(path[1]);

String year = String.*valueOf*(Integer.*parseInt*(path[0]));

**if**( month <= 6)

{

year=year.concat("Summer");

}

**else**

year=year.concat("Winter");

Double temperature;

Double Tmaxtemp = Double.*parseDouble*(path[4].toString());

Double Tmintemp = Double.*parseDouble*(path[5].toString());

temperature = (Tmaxtemp+Tmintemp)/2;

**if**(Tmaxtemp != ***MISSING*** && Tmintemp != ***MISSING***) {

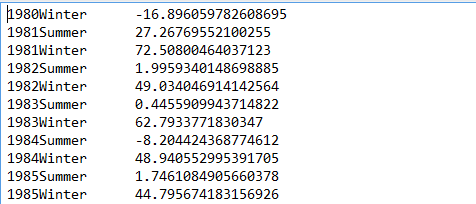
output.collect(**new** Text(year), **new** DoubleWritable(temperature));

}

}

}

**Output:**



**Time taken with 1 mapper 1 reducer:** 29882 milliseconds

**Datasets used for all above 3 codes:**

**Climate\_data.csv**: contains the sample data as below

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Year** | **Month** | **Day** | **Hour** | **MaxTemp** | **MinTemp** | **ID** | **Wind** | **Due** |  |  |
| 2005 | 1 | 1 | 0 | 22 | -6 | 10117 | 210 | 77 | 6 | 0 |

1. **Code to calculate average temperature year by year for each region.**

**public** **class** TemperatureMapper **extends** MapReduceBase **implements** Mapper<LongWritable, Text, Text, DoubleWritable>

{

**public** **static** **final** **int** ***MISSING*** = -9999;

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, DoubleWritable> output, Reporter reporter)

**throws** IOException

{

String line = value.toString();

String [] path = line.split(",");

String station = String.*valueOf*(Integer.*parseInt*(path[0]));

String year = String.*valueOf*(Integer.*parseInt*(path[2].substring(0,4)));

String station\_number = "Station";

station\_number = station\_number.concat(station);

String year\_no = "Year";

year\_no = station\_number.concat(year\_no);

year\_no = year\_no.concat(year);

Double temperature;

Double Tmaxtemp = Double.*parseDouble*(path[3].toString());

Double Tmintemp = Double.*parseDouble*(path[4].toString());

temperature = (Tmaxtemp+Tmintemp)/2;

**if**(Tmaxtemp != ***MISSING*** && Tmintemp != ***MISSING***) {

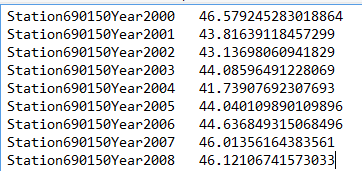
output.collect(**new** Text(year\_no), **new** DoubleWritable(temperature));

}

}

}

**Output:**



**Time taken with 1 mapper 1 reducer:** 29802 milliseconds

**Dataset used for above code:**

Climate\_regional.csv: contains below sample data

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STN--- | WBAN | YEARMODA | TEMP |  | DEWP |  | SLP |  | STP |  | VISIB |  | WDSP |  | MXSPD | GUST | MAX | MIN | PRCP | SNDP | FRSHTT |
| 690150 | 93121 | 20000601 | 83.3 | 23 | 40.1 | 23 | 1007.6 | 23 | 9999.9 |  |  | 23 | 8.2 | 23 | 15 | 18.1 | 98.6\* | 69.8\* | 0.00I | 999.9 | 0 |

**Reducer Code for all functionalities above:**

**public** **class** TemperatureReducer **extends** MapReduceBase

**implements** Reducer<Text, DoubleWritable, Text, DoubleWritable> {

**public** **void** reduce(Text key, Iterator<DoubleWritable> values,

OutputCollector<Text, DoubleWritable> output, Reporter reporter)

**throws** IOException {

**double** max\_temp = 0;

**int** count = 0;

**while** (values.hasNext()) {

max\_temp += values.next().get();

count+=1;

}

output.collect(key, **new** DoubleWritable(max\_temp/count));

}

}

**Time Taken:**

* 1 mapper 1 reducer : 29802 milliseconds
* 2 mapper 1 reducer : 29993 milliseconds
* 2 mapper 2 reducer : 29050 milliseconds
* 10 mapper 1 reducer : 37846 milliseconds
* 10 mapper 2 reducer: 37021 milliseconds