DISTRIBUTED COMPUTING: HADOOP PROGRAMMING

Lab Work
Hadoop as a MapReduce Platform and Java Classes as a Programming Framework

February 3rd, 2018

Jeremy Williams

Problem Statement

The goal of this lab work is to find out the average temperature in cloudera environment using python MapReduce based on the weather dataset provided. First python mapper is executed to collect the reduced dataset of year and temperate of that year. Once we have mapper results in hand, we invoke the reducer python to collect the temperature set against a year and then to sum of the temperature and then divided by count to get the average temperature of the year from the various temperature values of the year. This process was repeated for a Java implementation to implement the maximum temperature and then the average temperature.

Approach to Solution

The source codes were written in Python and Java (via pure python and java map-reduce techniques) for the results. A Linux-shell was used to initially run the mapper class then feed the mapper output to the reducer in both cases. The both codes have business logics that uses pure python and java map-reduce techniques.

Solution Description

As the codes are being run, from Linux-shell, the pipeline feeding of output for each case becomes available. The Execution logic is to run (in each case) the mapper first to get the reduced dataset to work on. Then find out and filter out only the valid data which are passing the validation criteria. Once the valid and filter dataset is available, the dataset will consist of **year** and **temperature** tuples.

Now the reduction mechanism is applied on the output of the above mapper output through 'reduceByKey' mechanism. This basically calculates the sum of the temperatures of a year first then divide by number of entries.

Data File

weather.txt

Sample data

```
0029029070999991901010106004+64333+023450FM-
```

12+000599999V0202701N015919999999N0000001N9-
00781+99999102001ADDGF108991999999999999999999

0029029070999991901010113004+64333+023450FM-
12+000599999V0202901N008219999999N0000001N9-
00721+99999102001ADDGF104991999999999999999999

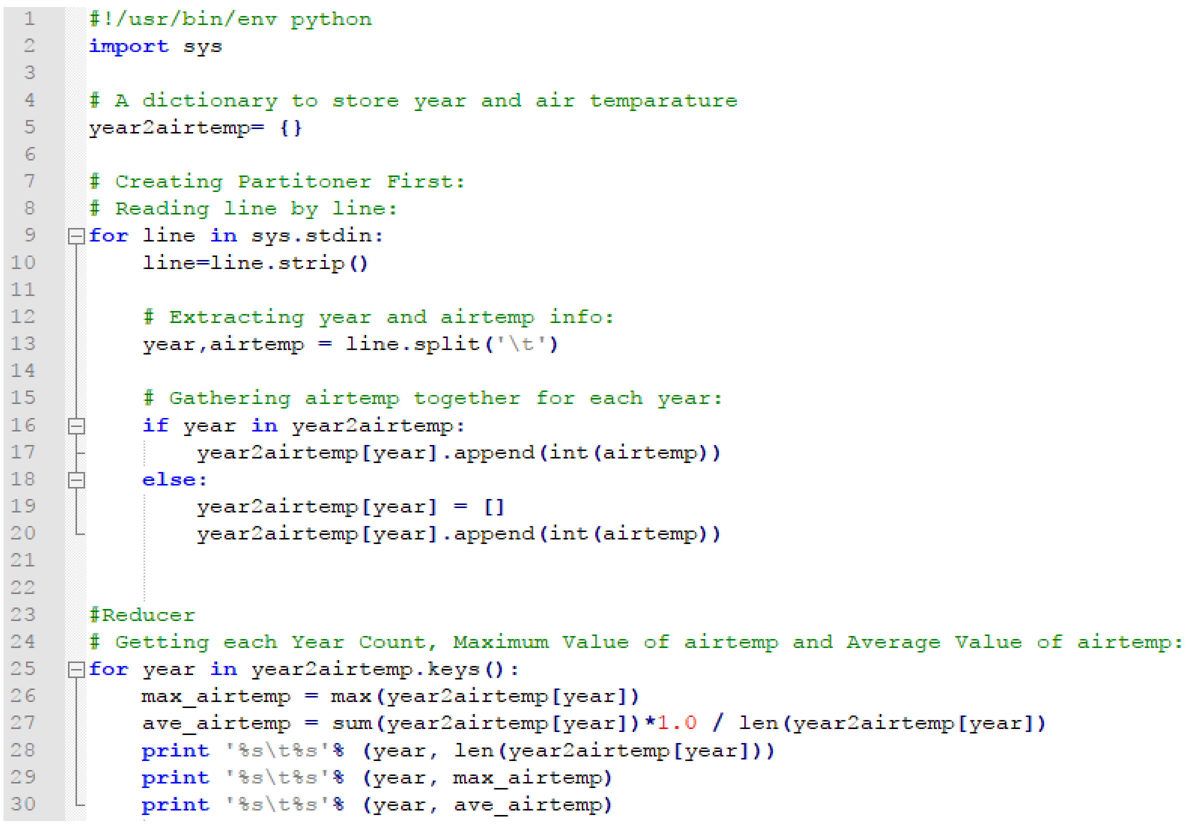
0029029070999991901010120004+64333+023450FM-
12+000599999V0209991C000019999999N0000001N9-
00941+99999102001ADDGF108991999999999999999999

0029029070999991901010206004+64333+023450FM-
12+000599999V0201801N008219999999N0000001N9-
00611+99999101831ADDGF108991999999999999999999

0029029070999991901010213004+64333+023450FM-
12+000599999V0201801N009819999999N0000001N9-
00561+99999101761ADDGF108991999999999999999999

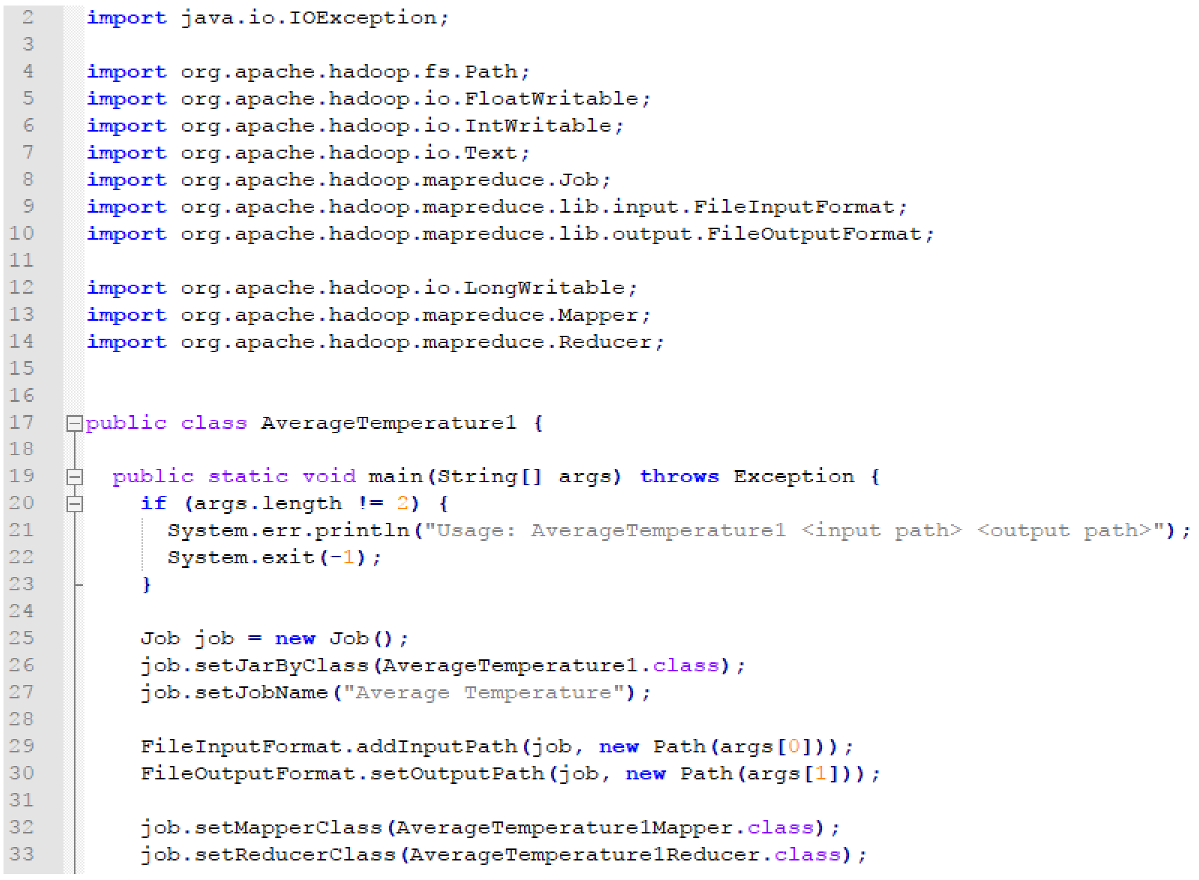
Final Codes

- Python

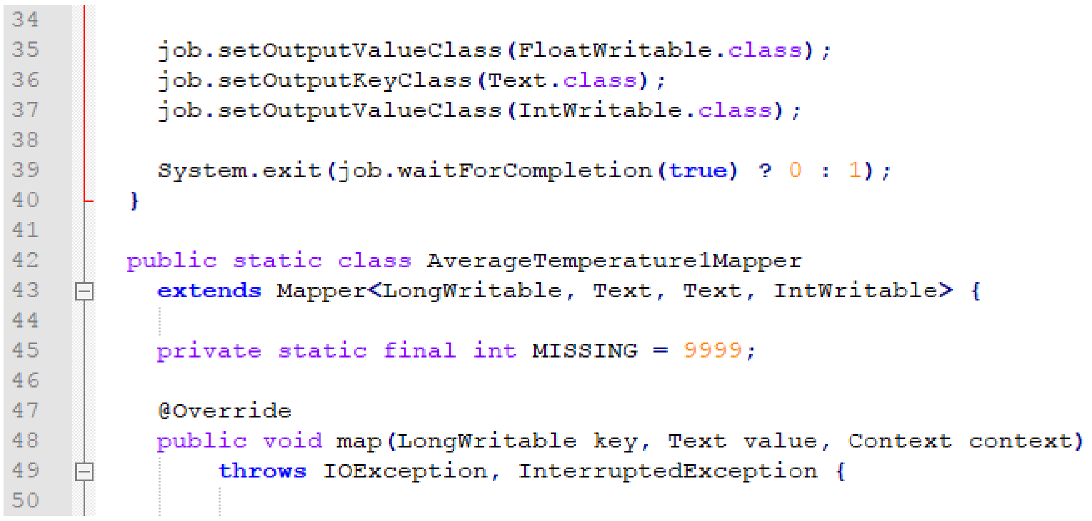


```python
#!/usr/bin/env python
import sys

# A dictionary to store year and air temparature
year2airtemp= {}

# Creating Partitoner First:
# Reading line by line:
for line in sys.stdin:
    line=line.strip()

    # Extracting year and airtemp info:
    year,airtemp = line.split('\t')

    # Gathering airtemp together for each year:
    if year in year2airtemp:
        year2airtemp[year].append(int(airtemp))
    else:
        year2airtemp[year] = []
        year2airtemp[year].append(int(airtemp))


#Reducer
# Getting each Year Count, Maximum Value of airtemp and Average Value of airtemp:
for year in year2airtemp.keys():
    max_airtemp = max(year2airtemp[year])
    ave_airtemp = sum(year2airtemp[year])*1.0 / len(year2airtemp[year])
    print '%s\t%s'% (year, len(year2airtemp[year]))
    print '%s\t%s'% (year, max_airtemp)
    print '%s\t%s'% (year, ave_airtemp)
```

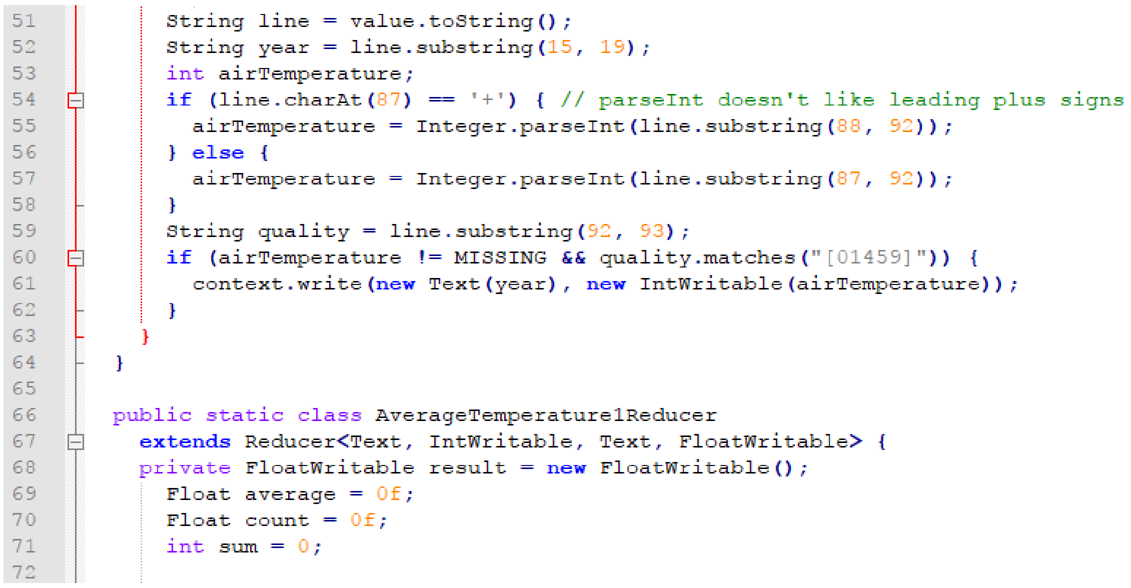
- Java



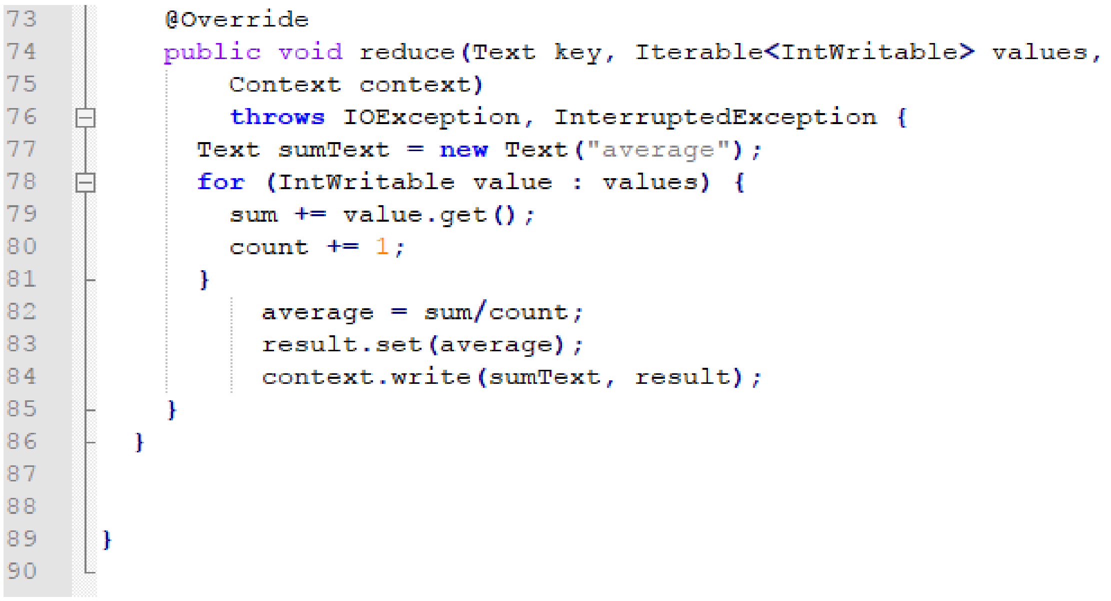
```java
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;


public class AverageTemperature1 {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: AverageTemperature1 <input path> <output path>");
      System.exit(-1);
    }

    Job job = new Job();
    job.setJarByClass(AverageTemperature1.class);
    job.setJobName("Average Temperature");

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(AverageTemperature1Mapper.class);
    job.setReducerClass(AverageTemperature1Reducer.class);
```



```java

    job.setOutputValueClass(FloatWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

  public static class AverageTemperature1Mapper
    extends Mapper<LongWritable, Text, Text, IntWritable> {

    private static final int MISSING = 9999;

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {

```



```
      String line = value.toString();
      String year = line.substring(15, 19);
      int airTemperature;
      if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
        airTemperature = Integer.parseInt(line.substring(88, 92));
      } else {
        airTemperature = Integer.parseInt(line.substring(87, 92));
      }
      String quality = line.substring(92, 93);
      if (airTemperature != MISSING && quality.matches("[01459]")) {
        context.write(new Text(year), new IntWritable(airTemperature));
      }
    }
  }

  public static class AverageTemperature1Reducer
    extends Reducer<Text, IntWritable, Text, FloatWritable> {
    private FloatWritable result = new FloatWritable();
      Float average = 0f;
      Float count = 0f;
      int sum = 0;

```



```
    @Override
    public void reduce(Text key, Iterable<IntWritable> values,
        Context context)
        throws IOException, InterruptedException {
      Text sumText = new Text("average");
      for (IntWritable value : values) {
        sum += value.get();
        count += 1;
      }
        average = sum/count;
        result.set(average);
        context.write(sumText, result);
    }
  }


}
```

Results/Outputs

-using Python

```
18/01/21 03:26:16 INFO streaming.StreamJob: Output directory: /hduser/output22
[cloudera@quickstart MaxTemp]$ hdfs dfs -ls /hduser/output22
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2018-01-21 03:26 /hduser/output22/_SUCCESS
-rw-r--r--   1 cloudera supergroup         38 2018-01-21 03:26 /hduser/output22/part-00000

[cloudera@quickstart MaxTemp]$ hdfs dfs -cat /hduser/output22/part-00000
```

```
1901    6564  ⇦ Temperature Count
1901    317   ⇦ Maximum Temperature
1901    46.6985070079 ⇦ Average Temperature
```

-using JAVA (Maximum and Average)

```
[cloudera@quickstart MaxTemp]$ hdfs dfs -ls /hduser/output8
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2018-01-07 12:45 /hduser/output8/_SUCCESS
-rw-r--r--   1 cloudera supergroup          9 2018-01-07 12:45 /hduser/output8/part-r-00000

[cloudera@quickstart MaxTemp]$ hdfs dfs -cat /hduser/output8/part-r-00000

1901    317   ⇦ Maximum Temperature


[cloudera@quickstart AverageTemp1]$ hdfs dfs -ls /hduser/output17
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2018-01-21 00:32 /hduser/output17/_SUCCESS
-rw-r--r--   1 cloudera supergroup         18 2018-01-21 00:32 /hduser/output17/part-r-00000

[cloudera@quickstart AverageTemp1]$ hdfs dfs -cat /hduser/output17/part-r-00000

 average   46.69850 ⇦ Average Temperature
```