Code Documentation: Temperature Analysis using MapReduce

Mehul Sharma(M24CSE013) Shivender Thapa(M24CSE023) Shreyank Jaiswal(M24CSE024)

September 30, 2024

1 Overview

This MapReduce-based program processes temperature data to identify hot and cold days from a dataset. The criteria used are:

- **Hot Day**: A day is considered hot if the maximum temperature exceeds 30°C.
- **Cold Day**: A day is considered cold if the minimum temperature falls below 15°C.

2 Modules

2.1 MaxTemperatureMapper (Mapper Class)

Functionality: This class reads temperature records, extracts the date, maximum temperature, and minimum temperature, and labels the day as ”Hot Day” or ”Cold Day” based on the temperature values.

Key Methods:

- **map()**: The method extracts relevant fields from each input record and emits key-value pairs for hot and cold days.

Input: Records containing date, maximum temperature, and minimum temperature.

Output: Emits key-value pairs where the key is a string indicating whether the day is hot or cold (e.g., ”Hot Day: YYYYMMDD”) and the value is the respective temperature.

```
class MaxTemperatureMapper(Mapper):
    MISSING = 9999

    def map(self, key, value, context):
        line = value.toString()
        if len(line) != 0:
            date = line[6:14]
            temp_Max = float(line[39:45].strip())
            temp_Min = float(line[47:53].strip())

            if temp_Max > 30.0:
                context.write(Text(f"Hot Day: {date}"), Text(str(temp_Max)))

            if temp_Min < 15.0:
                context.write(Text(f"Cold Day: {date}"), Text(str(temp_Min)))
```

2.2 MaxTemperatureReducer (Reducer Class)

Functionality: This class processes the key-value pairs emitted by the Mapper, aggregates the temperature data, and produces the final result.

Key Methods:

- **reduce()**: The method takes the temperature values corresponding to hot and cold days and aggregates them by simply passing the data forward (no further aggregation is necessary as each key represents a specific day).

Input: Key-value pairs of the date and temperature.
Output: Outputs key-value pairs as final results with date and temperature for hot and cold days.

```
class MaxTemperatureReducer(Reducer):
    def reduce(self, key, values, context):
        temperature = next(values).toString()
        context.write(key, Text(temperature))
```

2.3 Driver Code

Functionality: This is the main entry point of the program, responsible for configuring and running the MapReduce job. It sets the mapper and reducer classes, defines the input and output format, and handles file paths for input and output.

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "weather example");

    job.setJarByClass(MyMaxMin.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setMapperClass(MaxTemperatureMapper.class);
    job.setReducerClass(MaxTemperatureReducer.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

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

    OutputPath.getFileSystem(conf).delete(OutputPath);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

3 Map and Reduce Process

3.1 Map Phase

The `MaxTemperatureMapper` reads and processes each record from the dataset. For each record, it extracts the date, maximum temperature, and minimum temperature and applies the following logic:

- **Hot Day Condition**: If the maximum temperature is greater than 30°C, the mapper emits a key-value pair with the date and temperature.
- **Cold Day Condition**: If the minimum temperature is less than 15°C, the mapper emits a key-value pair with the date and temperature.

3.2 Reduce Phase

The `MaxTemperatureReducer` processes the key-value pairs generated by the mapper. It outputs the same key-value pairs as the final result because there is no need to aggregate multiple values (each key represents a unique day).

4 Input and Output

4.1 Input

A text file where each line represents a record of weather data. Relevant fields include the date, maximum temperature, and minimum temperature.

4.2 Output

A text file containing a list of hot and cold days along with the corresponding temperature values.

5 Test Data Example

Date	Max Temp (°C)	Min Temp (°C)
20240101	32.5	12.3
20240102	28.7	10.2
20240103	35.1	18.5
20240104	29.3	14.8

Table 1: Sample Test Data

6 Execution

To run the program, execute the following command in the Hadoop environment:

```
hadoop jar MyMaxMin.jar /path/to/input /path/to/output
```

7 Conclusion

This MapReduce implementation efficiently identifies hot and cold days from a large temperature dataset. The distributed nature of Hadoop allows the program to scale well with large datasets, providing quick insights into weather trends.