Design Document for Temperature Analysis Using MapReduce

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

September 30, 2024

1 Introduction

This article outlines the design and execution of a MapReduce-based approach for evaluating temperature data. The program analyzes a substantial dataset to determine hot and cold days according to established temperature thresholds.The program makes use of the Hadoop framework to efficiently distribute data processing over multiple nodes.

2 Objectives

This software uses MapReduce to find hot days (above 30°C) and cold days (below 15°C) in a dataset using MapReduce.

3 System Architecture

The system comprises two primary components:

- **Mapper:**Analyzes each data to identify hot and cold days based on temperature thresholds
- **Reducer:**Combines the identified hot and cold days to produce the final result.

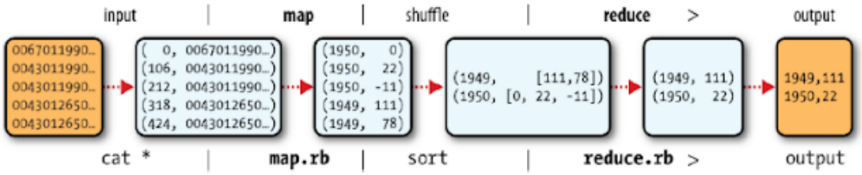

Figure 1: MapReduce Architecture for Temperature Analysis

4 Program Design

4.1 Mapper Design

The Mapper retrieves individual records from the collection, extracting the date, maximum temperature, and minimum temperature. If the maximum temperature surpasses 30°C, the day is classified as hot; conversely, if the minimum temperature falls below 15°C, it is classified as cold.

Pseudo Code:

```
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)))
```

4.2 Reducer Design

The Reducer aggregates the temperature data by processing the output of the Mapper and produces the final key-value pairs representing hot and cold days with their respective temperatures.

Pseudo Code:

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

5 Data Flow

The data flow for the MapReduce job is as follows:

- **Input:** A temperature data file comprising records that include the date, highest temperature, and minimum temperature.
- **Mapper Output:** Intermediate key-value pairs of dates categorized as hot or cold, accompanied by the corresponding temperatures.
- **Reducer Output:** Consolidated final values for warm and cool days.

6 Results

The results of the MapReduce job are as follows:

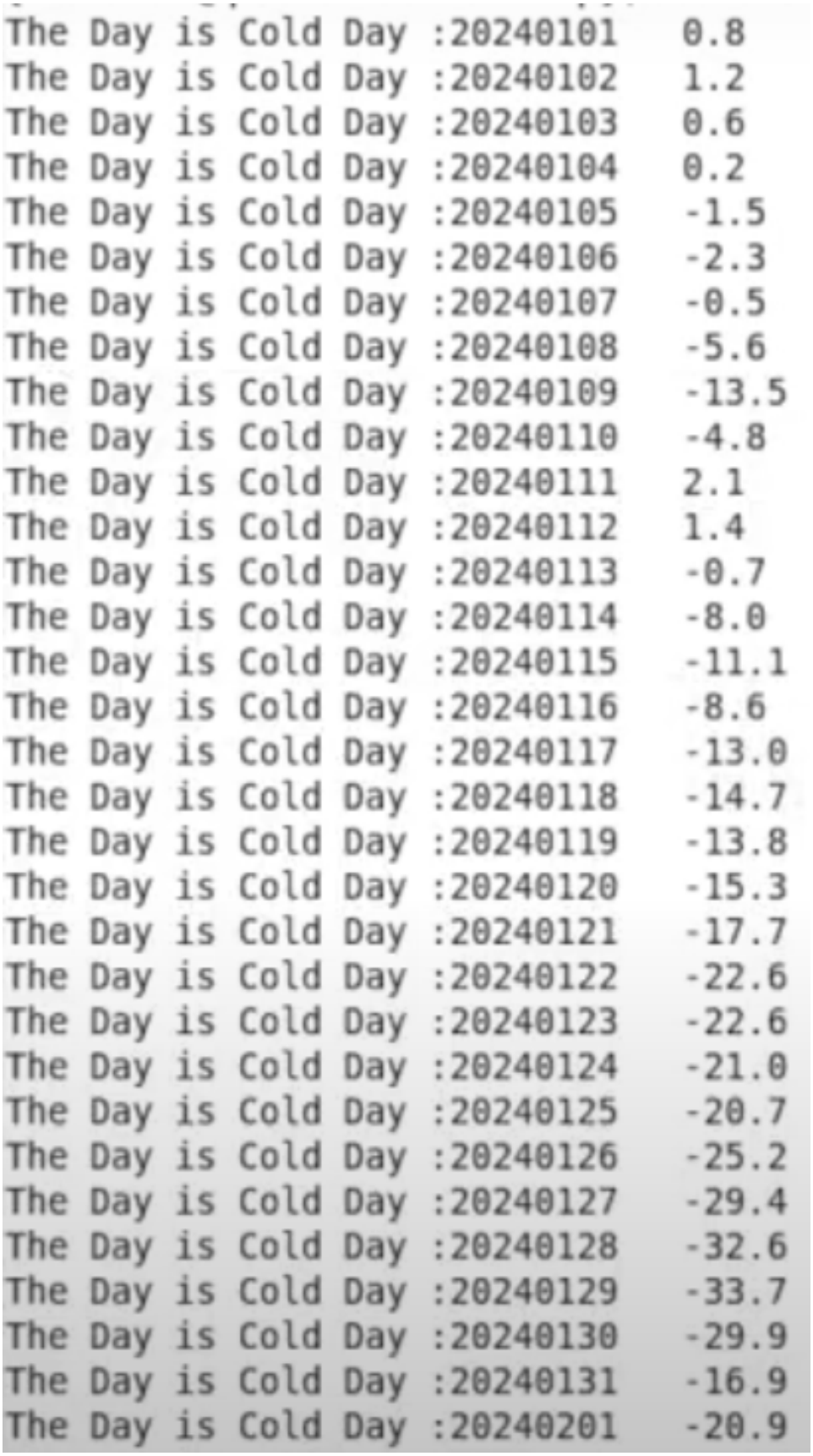

Figure 2: Hot and Cold Days Identified by MapReduce

7 Conclusion

The MapReduce-based temperature analysis efficiently discerned hot and cold days from the dataset by processing extensive data in a distributed framework. This method demonstrates the scalability and effectiveness of Hadoop for processing large datasets.