Assignment – 2

BIG DATA ANALYSIS

Hadoop MapReduce for Climate Data Analytics

NAME - SAHIL
ROLL NO. -107121086
ELECTRICAL AND ELECTRONICS ENGINEERING

Task 3

(b) Propose one more solution that the Map phase will be a clean-up phase that discards useless records, and for each day, it will calculate the temperature difference.

1. What will the key and values, output by the Map tasks be? What types will they be?

Solution: For the selected station and temperature types, it emits key/value pairs where the key is the date and the value is a concatenation of temperature type and temperature value.
Both output key and value are of Text (string) type.

2. Can the Mapper produce a key/value pair from a single input? How can we solve this issue?

Solution: Yes, in the context of Hadoop MapReduce, the Mapper produces key/value pairs from individual input records. In the Java code I provided, the TemperatureMapper class extends Mapper<LongWritable, Text, Text, Text>.
The map method in the TemperatureMapper class processes each line of input and emits key/value pairs using the context.write method:

context.write(new Text(date), new Text(tempType + "," + tempValue))

Here, new Text(date) is the output key, and new Text(tempType + "," + tempValue) is the output value. The map method is called for each input record, and for each record, the Mapper generates a key/value pair.
So, to answer your question, yes, the Mapper in Hadoop MapReduce produces key/value pairs from individual input records. Each record is transformed into one or more key/value pairs as specified by the logic in the map method.

3. For each day and weather station, the TMAX record always precedes the TMIN in NCDC's data, and we suppose that no split occurs between a TMAX and a TMIN record. By following this approach, what work will be left to the Reduce task?

Solution: Reduce Task will left with the calculation of the temperature difference between TMAX and TMIN for each day. Additionally, it has to normalizes the temperature values by dividing by 10 if they are greater than 10 or less than -10.

4. Reducer classes extend Hadoop's Reducer class. Read Hadoop's documentation for that class, in particular, what the default behaviour of its reduce() function is. What can you deduce from this?

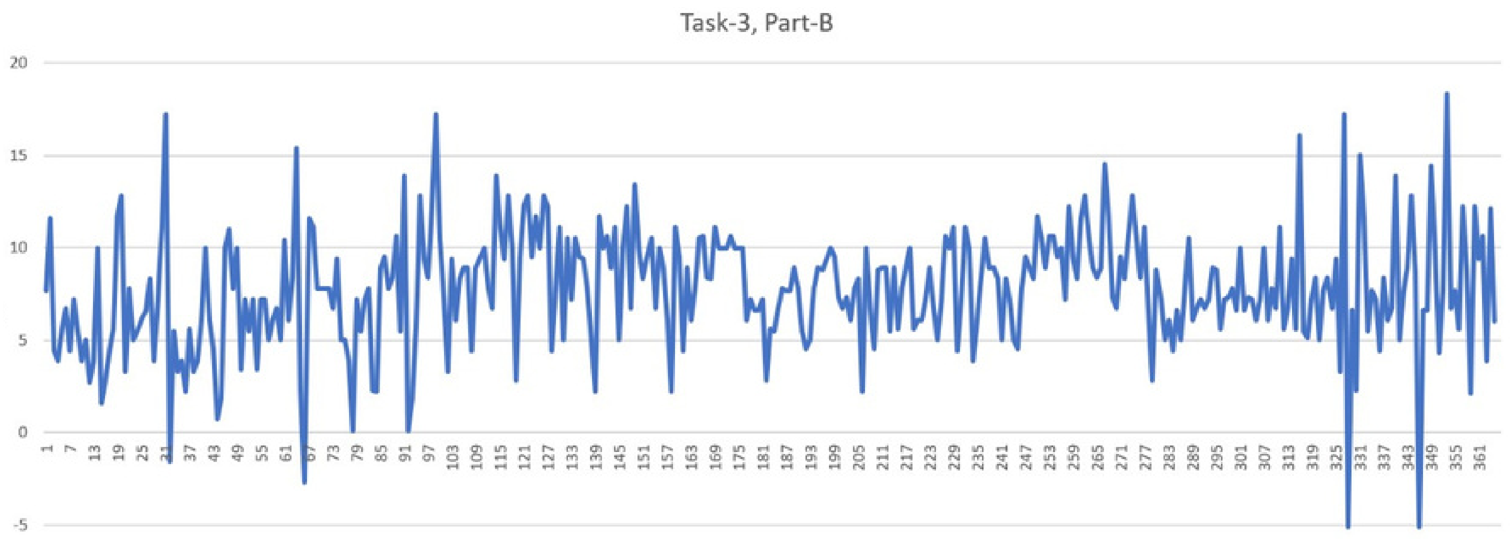
Solution: After going through the Hadoop's documentation, I came across several points given below :

1. Default Sorting:
The input to the reduce function is a sorted list of values for each key. This sorting is based on the natural ordering of the keys unless a custom key comparator is specified using **JobConf.setOutputKeyComparatorClass(Class)**.

2. **Iterable Input:**
   The reduce function receives an iterable (typically an iterator) over the values associated with a particular key. These values come from the outputs of all the mappers that share the same key.
3. **User-defined Logic:**
   The core logic of the reduce function is user-defined and must be implemented by the developer. This function is where the actual reduction or aggregation of values takes place based on the business logic defined by the user.
4. **Output:**
   The output of the reduce function is typically written to the FileSystem using the **OutputCollector.collect(WritableComparable, Writable)** method. The output key and value types are specified by the user.
5. **Unsorted Output:**
   The output of the Reducer is not automatically sorted. If a sorted output is desired, it's the responsibility of the developer to implement any additional sorting logic within the reduce function or consider post-processing steps.

In summary, the default behavior of the reduce function involves processing a sorted iterable of values for each key and applying user-defined logic to produce the final output. Sorting of the output is not automatic and needs to be explicitly handled if required by the application.

Plot of Output


Task-3, Part-B