Median and standard deviation[1]

Summarization pattern (Challenge 3_a)

For this exercise you have to prepare a report including:

- Your data collections comment describing what they represent.
- Your source codes: a file precising in a comment the compiling and execution commands.
- Propose a test battery and report execution measures.
- Compress everything and sent a .zip or a .tar to genoveva.vargas@gmail.com

1.1 Problem statement

Given a list of user's comments, determine the median and standard deviation of comment lengths per hour of day.

A median is the numerical value separating the lower and higher halves of a data set. This requires the data set to be complete, which in turn requires it to be shuffled. The data must also be sorted, which can present a barrier because MapReduce does not sort values.

A standard deviation shows how much variation exists in the data from the average, thus requiring the average to be discovered prior to reduction. The easiest way to per - form these operations involves copying the list of values into a temporary list in order to find the median or iterating over the set again to determine the standard deviation.

Attention: With large data sets, this implementation may result in Java heap space issues, because each value is copied into memory for every input group.

1.2 "Brutal" standard median deviation

Look at page 25 of the book "Map Reduce design patterns" and see the proposed `Map` and `Reduce` codes. Prepare a data collection of your choice and implement the solution. The book proposes StackOverflow or Wikipedia as data collection providers, you can choose any other collection that U like.

- Prepare collections of different sizes to run your tests trying to get to the limits of your solution.
- Make comparisons. Do not hesitate to prepare graphics.
- Explain and discuss why is it not possible (as stated in the book) to have a combiner. Support your arguments with examples.

1.3 Memory conscious standard median deviation

The following implementation is differentiated from the previous median and standard deviation example by reducing the memory footprint. Inserting every value into a list will result in many duplicate elements.

One way to get around this duplication is to keep a count of elements instead. For instance, instead of keeping a list of < 1, 1, 1, 1, 2, 2, 3, 4, 5, 5, 5 >, a sorted map of values to counts is kept: (1→4, 2→2, 3→1, 4→1, 5→3).

[1] This challenge is an example proposed in the book MapReduce design patterns, pp. 25.

- The core concept is the same: all the values are iterated through in the reduce phase and stored in an in-memory data structure.
- The data structure and how it is searched are all that has changed.

A map reduces the memory footprint drastically. Instead of having a list whose scaling is O(n) where n = number of comments, the number of key/value pairs in our map is O(max(m)) where m = maximum comment length. As an added bonus, a combiner can be used to help aggregate counts of comment lengths and output the map in a `Writable` object to be used later by the reducer.

Look at page 28 of the book "Map Reduce design patterns" and see the proposed `Map`, `Reduce` and `Combiner` codes.

- To test your solution, use the same collections of your previous solution.
- Explain changes in the reduction of the memory use complexity of this solution.
- Make comparisons with the previous. Do not hesitate to prepare graphics.
- Explain the optimization principle provided by the combiner that you implemented. Support your arguments with examples.