VE472 — Methods and tools for big data

Homework 3

Manuel — UM-JI (Summer 2021)

Reminders

- Write in a neat and legible handwriting or use LaTeX
- Clearly explain the reasoning process
- Write in a complete style (subject, verb, and object)
- Be critical on your results

Ex. 1 — *MapReduce*

In this exercise we write a MapReduce program to solve the second exercise from lab 2.

1. Write a `Map` class which extends the MapReduce `Mapper` class, extracts, and outputs pairs composed of a student ID and a grade.
   Hint: read the file by line and tokenize each of them using `StringUtils`.
2. Write a `Reduce` class which extends the MapReduce `Reducer` class, outputs pairs composed of a student ID and its highest grade.
   Hint: use `Iterable<Text>` to iterate over all the values of a given key.
3. Write a driver function write set all the necessary properties to configure the MapReduce job.
   Hint: specify what classes are to be used by the Mapper and Reducer, as well as where the input and output files are located.
4. Run the MapReduce program and compare the running time to the streaming approach used in the lab. Draw a table showing the comparison for various file sizes.

Ex. 2 — *Avro*

1. Explain the three ways or API styles into which Avro can be used in MapReduce, and when to apply each of them.
   Hint: the three approaches are (i) specific, (ii) generic, and (iii) reflect.
2. Use your MapReduce program from the previous exercise to process the Avro file produced in Homework 2 exercise 4.

Ex. 3 — *Bloom filters*

In general data should be filtered before running actions on it. For instance in Lab 2 exercise 2, one might want to retrieve the maximum grade for students whose ID ends with a three. An efficient way to achieve this is to run a preprocessing job to create a Bloom filter and filter out records in the mapper.

1. Describe what a Bloom filter is and how it works.
2. Using the `BloomFilter` class write a mapper which creates a Bloom filter.
   Hint: check Hadoop documentation for more details on the `BloomFilter` class.
3. Using `Iterable<BloomFilter>` combine all the Bloom filters together in the reducer and output the result into a serialized Avro file.