DASC 5433: Big Data Analytics

Assignment 2

Instructions.

1. Due date: Oct. 13.
2. Python version 3.7 or later is required.
3. Submit MRSimulator_DASC5433_Fall23_<LastName>_<ID>.py to Blackboard.

You will complete a back-end for a MapReduce system and test it on a couple of MapReduce jobs: word count, matrix multiply, and counting incomes by powers of 10. You will use trial_incomes.csv for Task 3. Data to test the completion of Tasks 1 and 2 are hard-coded into main.

Code Template. A template to be filled in with your code is provided in MRSimulator_DASC5433_Fall23_LastName_ID.py. Do not edit any blocks of code beyond those within the scope of "#[TODO]#". Existing `print` statements are designed to test your program during different points in the map-reduce process. You may add additional print statements during your internal tests but you must remove them before submission.

Code Familiarity.

1. Rename "LastName" and "ID" in the filename to your last name and your UHCL student id. Also, add them to the comment at the top of the code.
2. The code should run with "python3 MRSimulator_DASC5433_Fall23_LastName_ID.py trial_incomes.csv" but not produce results. Test that it does.
3. Examine the class `MyMRSimulator`. Start by reading the segments of the "`runSystem`" method to see the steps being run. Then examine each additional method.
4. Examine `def wordCountMR` for an example of a complete map and reduce function.
5. You will complete 2 segments of code in both the "`reduceTask`" and "`runSystem`", as well as write a map-reduce implementation of an algorithm as described below. Within the code, look for "#[TODO]#".

Task 1: ReduceTask. Complete the method "`reduceTask`" to perform the tasks of the reducer:

1. SEGMENT 1: sort such that all values for a given key are in a list for that key (i.e., (k1, v1), (k1, v2), (k1, v3) $\rightarrow$ {k1: [v1, v2, v3]})
2. SEGMENT 2: call `self.reduce(k, vs)` for each key, providing its list of values and add the results (if they exist) to the list variable "`namenode_fromR`"

Task 2: RunSystem. Complete the "`runSystem(self)`" method which divides the data into chunks and schedules the running of mapTasks and reduceTasks. There are two places to complete:

1. SEGMENT 2: Divide up the data into chunks according to `num_map_tasks`, and launch a map task per chunk.
2. SEGMENT 4: Send each key-value pair to its assigned reducer by placing it in the `to_reduce_tasks` dictionary.

Task 3: CountBy10PowersMR. Edit the "`map`" and "`reduce`" methods of "ApplyTopicsMR" to implement a map-reduce computation of counting the integers (representing incomes) by powers of 10. That is, for each integer round it down to its nearest power of 10 (e.g., 3 map to $1 = 10^0$; 30 would map to $10 = 10^1$. 87 would map to $10 = 10^1$; 870 would map to $100 = 10^2$, 100 would map to $100 = 10^2$, etc.). Your goal is to count the number of integers between each power of 10. Here is an example output after the final reduce: [(1, 446), (10, 258), (100, 133), (1000, 89), (10000, 50), (100000,24)].

Note. **Do not use self.data from the mappers or reducers: they need to work with the key values that they are provided. Your code should run in < 30 seconds across all tests.**[1]

[1]Many thanks to Dr. Schwartz