INP7079233 - BIG DATA COMPUTING 2023-2024 (prof. Pietracaprina and Silvestri)

Assignment of Homework 2 (DEADLINE: May 15, 23:59)

In this homework, you will run a Spark program on the CloudVeneto cluster. The program will test a modified version of the approximation strategy for outlier detection developed in Homework 1 where the distance parameter D is not provided in input by the user, but is set equal to the *radius* of a k-center clustering (for a suitable number K of clusters), that is, the maximum distance of a point from its closest center. In other words, the role of the input parameter D is replaced by K. This has two advantages: (1) a better control on the number of non-empty cells; and (2) the potential for a sharper analysis. The purpose of the homework is to assess the effectiveness of this strategy and to test the scalability of a MapReduce implementation when run on large datasets.

TASK for HW2:

1) **Modify of method/function MRApproxOutliers written for HW1** by removing the parameter K and the printing of the first K cells in non-decreasing order of cell size. (Please note that in HW2, a parameter \(K) is used outside MRApproxOutliers, but with a totally different meaning with respect to HW1.) Also fix bugs (if any) that have been pointed out in the correction of HW1.

2) **Write a method/function SequentialFFT** which implements the Farthest-First Traversal algorithm, through standard sequential code. SequentialFFT takes in input a set P of points and an integer parameter K, and must return a set C of K centers. Both p and C must be represented as lists (ArrayList in Java and list in Python). The implementation should run in $O(|P| \cdot K)$ time.

3) **Write a method/function MRFFT** which takes in input a set P of points, stored in an RDD, and an integer parameter K, and implements the following 3 round MapReduce algorithm:

- **Rounds 1 and 2** compute a set C of K centers, using the MR-FarthestFirstTraversal algorithm described in class. The coreset computed in Round 1, must be gathered in an ArrayList in Java, or a list in Python, and, in Round 2, the centers are obtained by running SequentialFFT on the coreset.
- **Round 3** computes and returns the radius R of the clustering induced by the centers, that is the maximum, over all points $x \in P$, of the distance $\mathrm{dist}(x, C)$. The radius R must be a float. To compute R you cannot download P into a local data structure, since it may be very large, and must keep it stored as an RDD. However, the set of centers C computed in Round 2, can be used as a global variable. To this purpose we ask you to copy C into a broadcast variable which can be accessed by the RDD methods that will be used to compute R.

MRFFT must compute and print, separately, the running time required by each of the above 3 rounds.

3) **Write a program GxxxHW2.java** (for Java users) **or GxxxHW2.py** (for Python users), where xxx is your 3-digit group number (e.g., 004 or 045), which receives in input, as command-line arguments, a path to the file storing the input points, and 3 integers M, K, L, and does the following:

- Prints the command-line arguments and stores M, K, L into suitable variables.
- Reads the input points into an RDD of strings (called **rawData**) and transforms it into an RDD of points (called **inputPoints**), represented as pairs of floats, subdivided into L partitions.
- Prints the total number of points.
- Executes MRFFT with parameters inputPoints and K and stores the returned radius into a float D
- Executes MRApproxOutliers, modified as described above, with parameters inputPoints, D, M.

This file (**TO BE ADDED**) shows how to format your output. *Make sure that your program complies with this format.*

4) Test and debug your program in local mode on your PC to make sure that it runs correctly. The program must be stand-alone in the sense that it should run without requiring additional files.

5) Test your program on the cluster using the datasets which have been preloaded in the HDFS available in the cluster. Use various configurations of parameters and, in particular, fill the table given in this word file (**TO BE ADDED**) with the results of the experiments specified in the file.

WHEN USING THE CLUSTER, YOU MUST STRICTLY FOLLOW THESE RULES:

- To avoid congestion, groups with even (resp., odd) group number must use the clusters in even (resp., odd) days.
- Do not run several instances of your program at once.
- Do not use more than 16 executors.
- Try your program on a smaller dataset first.
- Remember that if your program is stuck for more than 1 hour, its execution will be automatically stopped by the system.

SUBMISSION INSTRUCTIONS. Each group must submit a zipped folder GxxxHW2.zip, where xxx is your group number. The folder must contain the program (GxxxHW2.java or GxxxHW2.py) and a file GxxxHW2table.docx with the aforementioned table. Only one student per group must do the submission using the link provided in the Homework 2 section. Make sure that your code is free from compiling/run-time errors and that you comply with the specification, otherwise your grade will be penalized.

If you have questions about the assignment, contact the teaching assistants (TAs) by email to bdc-course@dei.unipd.it . The subject of the email must be "HW2 - Group xxx", where xxx is your group number. If needed, a zoom meeting between the TAs and the group will be organized.

Last modified: Monday, 22 April 2024, 10:44 PM

◄ User Guide for the Cluster provided by Cloud Veneto

Jump to...

Submission form for HW2 ►