Assignment
Map-Combine-Reduce

Design Lab
02nd February 2024

Important Instructions

1. **Submission Rule:** All deliverables must be compressed as **.tar.gz** (gzip) and named "**<RollNo>.tar.gz"**. Strictly adhere to this naming convention. Submissions not following the above guidelines will attract penalties.
2. **Code error:** If your code doesn't run or gives error while running, you will be awarded with zero mark. Your code must run correctly on **a linux machine**.
3. **Plagiarism Rule:** If your code matches (above a particular threshold) with another student's code, all those students whose codes match will be awarded with zero marks without any evaluation. Therefore, it is your responsibility to ensure that neither you copy anyone's code nor anyone is able to copy yours.
4. The mapper routine can pass over the dayx.txt file only once. Python dictionaries should not be used in the mapper routine. However, you can use arrays or dictionaries of size up to 300 in the reducer only. These must be limited to one dimension only. The combiner and reducer will have access to only the stream of key-value pairs but nothing else. Any violation will attract a penalty upto 100% of the marks assigned.

This assignment is on map-combine-reduce, which is a distributed and scalable way of extracting/mining required information from multiple datasets stored on multiple servers. Follow the tutorial to understand how you can design mapper and reducer for specific queries/operations.

Dataset

We will be using a similar dataset on a Facebook (FB) network with only **300** users. The data contains 10 files containing logs of chat sessions from 1st Feb to 10th Feb 2021. You have to extract required information from this data. Below is a sample of data available in the file.

(**day3.txt**)

```
276,107
284,68
98,246
97,49
292,58
119,213
229,156
```

where

- For example: each line in the file day3.txt represents a chat session between two users (node-ids separated by comma) on 3rd Feb 2021. Session logs appearing earlier represent earlier chat sessions and later logs represent later chat sessions.

Sample code to read the files

```
fp=open("day3.txt")
for line in fp:
    l_arr=line.split(",")
    node1=l_arr[0]
    node2=l_arr[1]
```

Steps (marks assigned to a step is conditional on the results of previous step(s))

The queries are to be implemented in the mapper (also combiners wherever asked) and reducer. You need to implement the following queries in this assignment.

1. **Finding the strongly connected network:** You have been given the dataset with logs of chat sessions between the users on FB. Rather than relying on the actual friendship network (as we used in the last assignment), this time, we want to find the strongly connected nodes using how many times a pair of nodes chatted in the 10 days. If a pair of nodes have chatted at least 10 times in the 10 days, then only you consider them to be strongly connected and have an edge in between them. Now find all such strongly connected pairs of nodes to create the chat network. As there are 10 different files, you will have to write mapper, combiner, and reducer routines. Save the network in "network.txt".
   (**deliverables**: mapper1.py, combiner1.py, reducer1.py, network.txt) **[30]**
   (**P.S.-** *The combiner must behave as a mini-reducer at mapper level; i.e. each day's file should go through mapper and combiner individually. The combiner shouldn't just be receiving and sending the same key-value pairs.*)

   (python mapper1.py file1 | sort | python combiner1.py & python mapper1.py file2 | sort | python combiner1.py & ... | sort | python reducer1.py)

2. Shuffle and split the computed **network.txt** in the previous question into K (for K in 5 to 20 at increments of 5) chunks. Compute the answers to the following queries in mapper-combiner-reducer fashion for each K and log the time taken to compute the answers for each of the query. Also compute the time taken to compute the answers for each of the query using entire **network.txt** using map-reduce only once along with the time taken to compute the answer for the query.
   a. Find the node which has chatted the most with other nodes in the network
   b. Find the node pair(s) that did not chat in this network.
   c. Find those nodes which could be ***potential whisperers*** –nodes that are *articulation points* in the network.

   (**Hint**: *Try to build a graph representation in the reducer*) **[15+15+30+10]**

   For each query, for each value of K, then compute the speedup obtained and plot the speedup obtained. On x-axis keep values of K, and on y-axis keep the speedup achieved.

   Deliverables:
   1. 3 mappers, 3 reducers, 3 combiners, 3 Readme.txt explaining your approach..
   2. 1 mapper, 1 reducer per query
   3. 1 report containing 3 plots of speedup per query.