Assignment 7
Map-Combine-Reduce

Computing Lab (II)
17th Feb 2021

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.

Tutorial on combine

We learned about map-reduce in the last week. This week we are looking into combiners too. In many real world scenarios, the mapper has to send too many key-value pairs, which then reach the Hadoop/MapReduce management system to be grouped and sorted. Having too many key-value pairs can become a bottleneck for the system. Instead if we introduce some form of grouping in the mapping phase itself, then the load on the whole system will be significantly reduced. So, we plan to use combiners (mini-reducers at local mapping level wherever possible) in the mapping phase before passing the key-value pairs to be shuffled and sorted.

Combiner Reference:

https://www.tutorialspoint.com/map_reduce/map_reduce_combiners.htm

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** (mapper+combiner+reducer)**:** 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 30 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)
   (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) **[35]**

2. **Finding the number of mutual friends:** Here, you want to find out how many mutual friends each pair of nodes have in the chat network created in the last question. Write mapper and reducer routines. Save the results as <node-1,node-2,#mutualFriends> in “friends.txt”. (**deliverables**: mapper2.py, reducer2.py, friends.txt)
   (Hint: If A is the adjacency matrix for the network, A^2 will have the number of mutual friends. The mapper can take the network.txt from the last question as input.) **[30]**

3. **Finding the number of triangles in the network**: If there are edges like (v_1,v_2), (v_2,v_3), and (v_3,v_1) then it forms a triangle in the network. Here you have to find out the total number of triangles passing through each node. Write mapper and reducer routines for this, and save the results as <node,#trianglesThrough-node> in “triangles.txt”.
   (**deliverables**: mapper3.py, reducer3.py, triangles.txt)
   (Hint: Diagonal elements of A^3 can be used here. The mapper3 can access network.txt and friends.txt here) **[25]**

Evaluation Scheme: Results: 90 marks, Coding Style: 10 marks

Important Instructions

1. **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 10 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.

2. **Submission Rule:** All deliverables must be compressed as **.tar.gz** (gzip) and named "**A7_<RollNo>.tar.gz"**. Strictly adhere to this naming convention. Submissions not following the above guidelines will attract penalties.
3. **Makefile:** The folder **must** have it's own Makefile to execute the routines one after another. You can find a relevant tutorial here (https://opensource.com/article/18/8/what-how-makefile).
   Note: "sort" behaves differently in different environments. Ensure that you use sort in a way which works properly on a linux machine.
4. **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**.
5. **Plagiarism Rule:** If your code matches (more than 50%) with another student's code, all those students whose codes match will be awarded with zero marks (may be with -ve marks too depending on the situation) without any evaluation. Therefore, it is your responsibility to ensure that neither you copy anyone's code nor anyone is able to copy yours.