Big Data 18CS322

Assignment 2

Page Rank Algorithm implementation with Map Reduce

1. Assignment Objectives and Outcomes

a. The objective of this assignment is for the students to run iterative processing with Map Reduce and learn how the Map Reduce algorithm works..

b. At the end of this assignment, the student will be able to write and debug Page Rank code on Map Reduce.

2. Ethical practices

Please submit the original code only. You can discuss your approach with your friends but you must write original code. All solutions must be submitted through the portal. We will perform a plagiarism check on the code.

3. The Dataset:

a. Will be shared on Forums soon

4. Software/Languages to be used:

a. Python (Version **3.6**) (Use Hadoop Streaming)

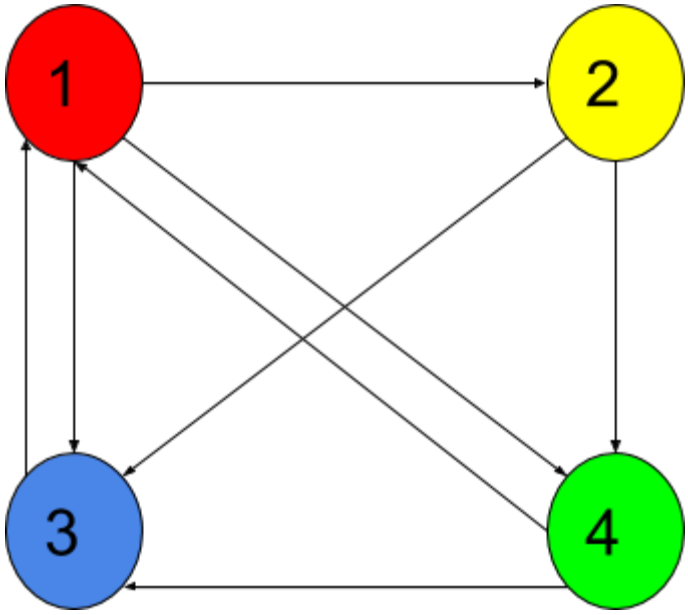
b. Please make sure to use Hadoop Version **3.2.0** only.

5. Marks:

a. 10 (Scaled down to 5)

b. Each Task is for 5 marks

6. Tasks:



a. Task 1 - Convert input file to adjacency list using map reduce

i. Description

- Store the input file on HDFS
- The input will be uploaded as a .txt file on Forums. It is of the format:
  `from_node_id  to_node_id`
- Write Map and Reduce functions to read the input text file and generate two files representing the adjacency list and initial ranks
- The reducer's output will be of the format:
  `from_node_id  list_of_adj_nodes`
- These initial page ranks should be stored in a new file called "v" **(format to be strictly followed):**
  `node, pagerank`

  The values are comma separated and newline delimited

ii. Comments:

- The adjacency list should also be **written on HDFS** in a file called "adj_list", and the page rank vector should be stored locally in a file called "v"
- Refer to the example for the format to create these files.

- **Important: Never load the entire input text file or the adjacency list into your memory!**
- The input list may not be sorted, but it will be grouped by "from_node_id"
- The nodes may not always be numeric

iii. Example:

- Input list:

```
2	3
2	4
1	2
1	3
1	4
3	1
4	3
4	1
```

- "v" file

```
1, 1
2, 1
3, 1
4, 1
```

- "adj_list" file - Can be modified as per your requirements for Task 2:

```
1	[2, 3, 4]
2	[3, 4]
3	[1]
4	[3, 1]
```

iv. Explanation

- The initial page rank (stored in "v" file) is initially 1 for all nodes

b. Task 2 - Write mapper and reducer to calculate page rank until convergence

i. Description:

- Mapper reads input from "adj_list" and "v" and for each node `i`, it emits key,value pairs
- Reducer groups by key and sums up the values to calculate the new page ranks using the formula:
  `0.15 + 0.85 * (sum {contribution of every node that links to it} )`
- `contribution of every node` is calculated as:
  `pagerank of node/(number of outgoing links from node)`

- These new page ranks should be stored in a new file called "v1" **(format to be strictly followed):**

```
node, new_pagerank
```

The values are comma separated and newline delimited

ii. Comments:

- We will provide a bash script on Forums that will perform the following operations:
  a. Mapper reads "adj_list" and "v" and writes output to a file "m_out"
  b. Reducer reads "m_out" and writes output to "v1"
  c. If values of "v" and "v1" are nearly similar (i.e, has reached convergence):
     Exit
     Else:
     Delete "v" and rename "v1" to "v"
     Redo from step a
- Reaching convergence means that the difference between page ranks for every node should be < n
- The value of n will be decided by the bash script - it will vary across different test cases

iii. Example (continued) -

- **Values mentioned below are only for the first iteration**
- Reducer output "v1" **(format to be strictly followed)**

```
1, 1.425
2, 0.4305
3, 1.2805
4, 0.855
```

iv. Explanation:

- The above input adjacency list can be converted to the following matrix (M) where M_{ij} is the contribution of **node `j` to node `i`**

	node 1	node 2	node 3	node 4
node 1	0	0	1	0.5
node 2	0.33	0	0	0
node 3	0.33	0.5	0	0.5
node 4	0.33	0.5	0	0

- Reducer output for node 1:

  Node 1 receives contribution values 1 and 0.5 (from nodes 3 and 4 respectively)
  `1, (0.15 + 0.85(1+ 0.5) = 1.425)`

  Node 2 receives contribution value 0.33 from Node 1 only
  `2, (0.15 + 0.85(0.33) = 0.4305)`

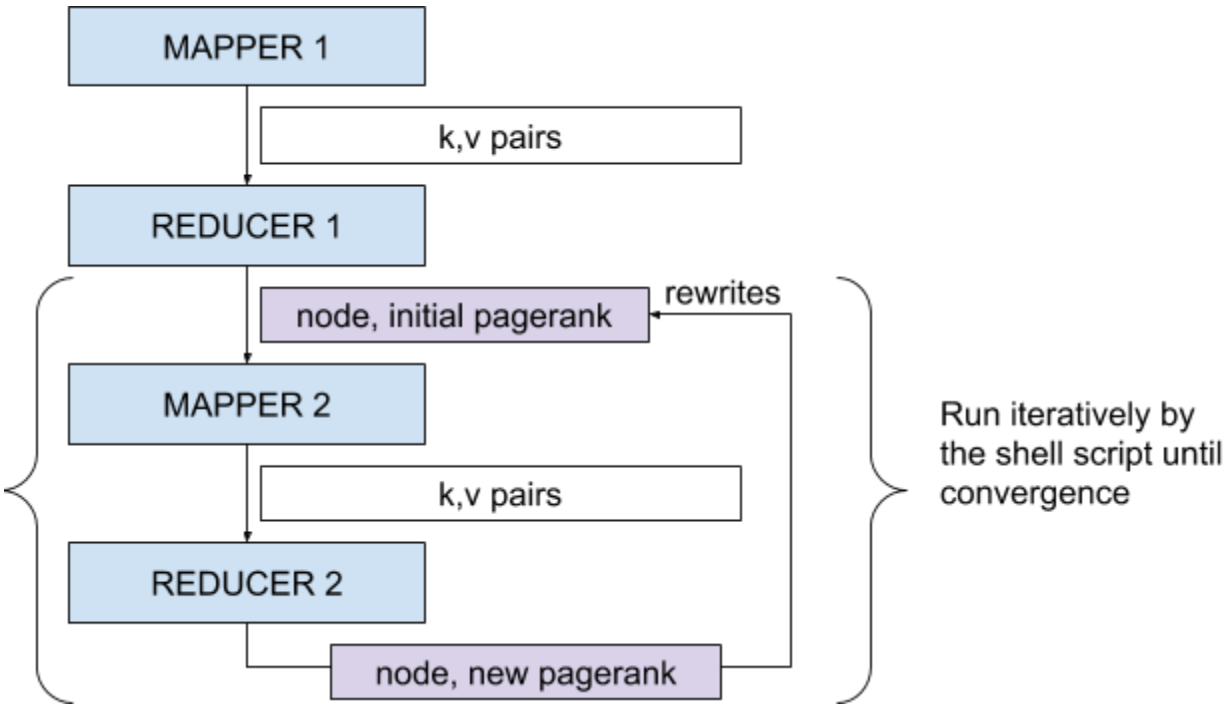
- Follow similar logic for other nodes

v. Take note of:

- Number of iterations taken for convergence
- Performance measure of total time taken (in ms)

c. Write the output of the mapper and reducer for each question in the report.
d. Submit a one page report based on the template and answer the questions on the report.

7. Summarization of Page Rank program:



8. Keep an eye on Forums for:

a. Dataset

b. Bash script to automate your MR functions

c. Commands used for execution at the backend

d. Report Template

e. Submission Date and Link

9. Resources:

a. **Anchor's slides for Matrix multiplication and Page Rank using Map-Reduce**

b. https://github.com/zHaytam/PageRank

c. https://youtu.be/9e3gelYFOF4