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)

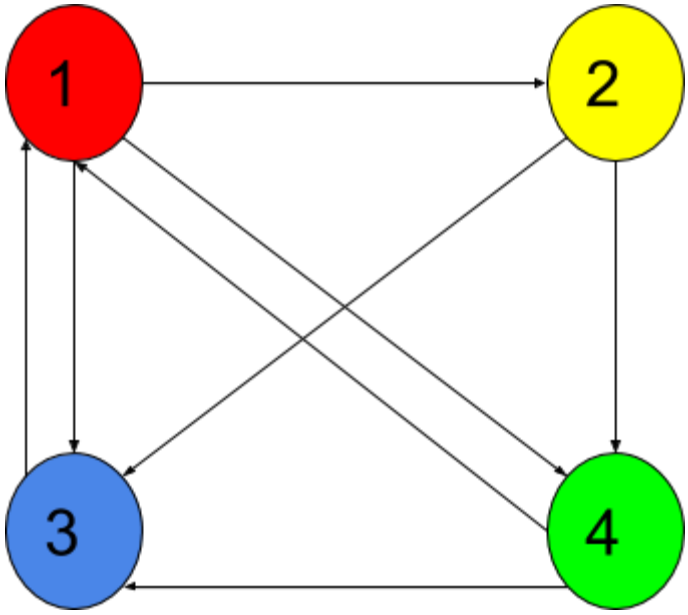
b. Please make sure to use Hadoop Version **3.2.0** only.

5. Marks:

a. 10 (Scaled down to 5)

b. Each Task is for 5 marks

6. Tasks:



a. Task 1 - Convert adjacency list to sparse matrix

i. Description

- The input adjacency list will be uploaded as a .txt file on Forums
- Convert input adjacency list to a sparse matrix and create initial page rank vector
- Store the sparse matrix on HDFS and page rank vector locally

ii. Comments:

- The sparse matrix should be **stored on HDFS** in a file called "sparse_matrix", 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.
- You can assume that the initial page rank vector can fit into memory at any given point.
- **Important: Never load the entire sparse matrix into your memory!**
- For Python - use the subprocess module to write to HDFS

iii. Example:

- Input adjacency list:

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

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

- "v" file

```
1, 0.25
2, 0.25
3, 0.25
4, 0.25
```

- "sparse_matrix" file - Can be implemented as per your requirements for Task 2

iv. Explanation

- The above mentioned adjacency list can be converted to the following matrix where
  value(i,j) = m_{ij} transition probability from 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

- The initial page rank (stored in "v" file) is calculated as:
  1/(no. of unique nodes)

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

i. Description:

- Mapper reads input from "sparse_matrix" and "v" and emits key,value pairs of the format:
  $(i, m_{ij}v_j)$
  where i,j represent the row and column of the sparse matrix and m_{ij} represents the value in row i, column j of the sparse matrix.

- Reducer groups by key and sums up the values to calculate the new page ranks. These new page ranks should be stored locally in a new file called "v1"

ii. Comments:

- We will provide a bash script on Forums that will perform the following operations:

  a. Mapper reads "sparse_matrix" 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

- The value to check for convergence 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**
- “m_out”

```
1, 0.25
1, 0.125
2, 0.08333333333333333
3, 0.08333333333333333
3, 0.125
3, 0.125
4, 0.08333333333333333
4, 0.125
```

- “v1”

```
1, 0.375
2, 0.08333333333333333
3, 0.3333333333333333
4, 0.20833333333333331
```

iv. Explanation:

- Mapper output for node 1:

```
1, (1* 0.25 =0.25)
1, (0.5*0.25 = 0.125)
```

- Reducer output for node 1:

```
1, (0.25 + 0.125 = 0.375)
```

- 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. 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

8. Resources:

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

b. https://youtu.be/9e3geIYFOF4