CS 5300
Project 2: Fast Convergence PageRank in Hadoop
Team member: Xiaoan Yan(xy254) Hengyi Huang(hh548) Yukun Jiang(yj297)

NetID used: xy254

How to Test the Code (on a local machine):
First unzip our file.
Then get into "code_on_local" directory:
"cd <path to solution directory>/code_on_local"
Then
"./preprocessing"
This command downloads "nodes.txt", "blocks.txt" and "edges.txt" and put them under "raw_input" directory

Then
" make &>log.txt"
This command starts the main program and write log output into log.txt

Error
"single_error" includes residual errors for single node implementation while "block_error" includes errors for blocked PR implementation.

Extra Credit
First of all get into "code_on_local" directory:
"cd <path to solution directory>/code_on_local"
Then:
"python block_hash.py <path to nodes.txt> nodes.txt"
Then we will see a new nodes.txt with random partition.

When we try this new nodes.txt file, the program does not converge in the first eight iterations.

Slide:
Do use the browser open up "presentation_slide.html" under project directory.

Architecture:
The main skeleton is inherited from the page rank assignment from the database class in last semester.

TrustMapper:
TrustMapper is triggered when hadoop finishes reading data from input data file. It traverse all the nodes in hadoop filesystem and emit nodes or page rank weights. The input is a key-value pair.

The key is the node ID while value is node object. First of all it calculates the page rank weight of each out-going link. Then it writes both node itself along with the weights of out-going links into the context.

TrustReducer:
The reducer basically follows the algorithm on the assignment writeup. It zero out all the PR values in each in-block node. Then it collects both inner-block and outer-block PR weights and push them into two hashtables. Finally, it updates the PR value for each node in the current block. The alogrithm repeats until a threshhold value is reached and normally it takes four or five iterations.

How to deploy .jar on EMR service on AWS:
First of all you have to do

"cd <path to code_on_local directory>

Then run

"./preprocessing"

and get the "input.txt" under "input" directory. We will be using this input.txt as our input file.

Set the input and output path in PageRank.java. We already implement that. There are two ways to set the address. First one is to use hard code in PageRank.java, for example "s3n://CornellCS5200_Project2/input". This way is easy for testing code. The other method is to set the input/output path in the arguments list in EMR console, and use parameters like "args[0]", "args[1]" to refer the paths.

The second thing should be noticed is the output file name and its route. We also have already implemented this part. The most efficient approach is to use the "TextOutputFormat.class" provided by Hadoop to be the output format class set by job.

After that what we need to do is just export the .jar file out of java files. And we need to upload this file as well as the input file(after processed by Python script) to the S3. Then we create a cluster, set the log file route, select the number of instances, input the jar file route, and write the main class(PageRank), input and output directory(in S3 format) in the Argument box. Finally we can create and wait to see the results in S3.