Project2: PageRank in Hadoop

Haoyu Jia(hj342)
Ke Wang(kw427)
Yan Huang(yh553)

1. Data preprocess:

Netid we used in the project: yh553
rejectMin: 0.99*0.35
rejectLimit: 0.99*0.35+0.01

number of edges selected: 7524160

2. Overall Structure:

Our code in this project composed of three parts: one for map, one for reduce and one for main.
Main module -- PageRank.java:
Main module would start a MapReduce job for each pass, and a MapReduce job would call "map" and "reduce" functions in map and reduce module.
Mapper -- PageRankCalcuateMap.java:
"map" function would parse the preprocessed input and produce output for "reduce".
Reducer -- PageRankCalculateReduce.java:
"reduce" function would calculate the new PageRank and other information and produce output for "Map" as input of the next pass. The detail of input/output format is described in the following sections.
Hadoop Count -- MyCounter.java:
When the specified number of passes has completed, the result information, e.g: average residual, average number of iterations, node ID with highest rank value, etc. would be printed in stdout. result information is passed from Reducer to Main by Hadoop counter.

Note: The total number of nodes is hard-coded in the program.

3. Single Node PageRank:

Mapper Input Format / Reducer Output Format:
<from_node, pagerank, to_node >

Mapper Output Format / Reducer Input Format:
1. <node_id: | lists of nodes>
2. <node_id: ! old PR>
3. If out-degree > 0: <node_id: PR/deg > // to_node, PR(from_node)/deg

Average residual errors in each MapReduce passes:
Pass 0: 2.3387482810589146
Pass 1: 0.3228494055864454
Pass 2: 0.19202766059571238
Pass 3: 0.09402209335551567
Pass 4: 0.06271717639332779

It is far from converged.

4. Blocked PageRank (Jacobi):

Mapper Input Format / Reducer Output Format:
<u_id+block_id "\t" PR "\t" v0_id+block_id0,v1_id1+block_id1,.........>
(from_node, pagerank, list of to_nodes)

Mapper Output Format / Reducer Input Format:
1. Current Rank Value: <Block_id : ! u_id "\t" PR>
2. Links: <Block_id : | u_id "\t" v0_id0+block_id0,v1_id1+block_id1 , …....>
3. <Block_id0 : v0_id "\t" PR "\t" from_u_id+Block_id_i "\t" degree_of_u >
//sink node, no output, degree = 0
<Block_id1 : v1_id "\t" PR "\t" from_u_id+Block_id_i "\t" degree_of_u >

What if node u has no out-links ("Sink node")?
Add an edge to itself with degree "0" in Map; Discard the PageRank from a source with "0" degree in Reduce. This handling method is used for both per-node PageRank and blocked PageRank.

Average number of iterations per block performed by Reducer:
Pass 0: 17
Pass 1: 7
Pass 2: 6
Pass 3: 4
Pass 4: 3
Pass 5: 1

Global Average Residual Error for each pass:
Pass 0: 2.815
Pass 1: 0.03818
Pass 2: 0.02395
Pass 3: 0.009886

Pass 4: 0.003847
Pass 5: 0.0009584

PageRank value for highest-numbered nodes in each block:

```
Highest Rank Node in Block 0 is 8354
Highest Rank Node in Block 1 is 16563
Highest Rank Node in Block 2 is 25567
Highest Rank Node in Block 3 is 33067
Highest Rank Node in Block 4 is 40655
Highest Rank Node in Block 5 is 50463
Highest Rank Node in Block 6 is 60843
Highest Rank Node in Block 7 is 70647
Highest Rank Node in Block 8 is 80124
Highest Rank Node in Block 9 is 94237
Highest Rank Node in Block 10 is 100696
Highest Rank Node in Block 11 is 110643
Highest Rank Node in Block 12 is 129662
Highest Rank Node in Block 13 is 139525
Highest Rank Node in Block 14 is 140574
Highest Rank Node in Block 15 is 161328
Highest Rank Node in Block 16 is 167683
Highest Rank Node in Block 17 is 174682
Highest Rank Node in Block 18 is 184001
Highest Rank Node in Block 19 is 192982
Highest Rank Node in Block 20 is 211611
Highest Rank Node in Block 21 is 222760
Highest Rank Node in Block 22 is 232579
Highest Rank Node in Block 23 is 237032
Highest Rank Node in Block 24 is 245877
Highest Rank Node in Block 25 is 258804
Highest Rank Node in Block 26 is 265974
Highest Rank Node in Block 27 is 280153
Highest Rank Node in Block 28 is 289833
Highest Rank Node in Block 29 is 297951
Highest Rank Node in Block 30 is 309302
Highest Rank Node in Block 31 is 319654
Highest Rank Node in Block 32 is 323550
Highest Rank Node in Block 33 is 343322
Highest Rank Node in Block 34 is 345482
Highest Rank Node in Block 35 is 355915
Highest Rank Node in Block 36 is 370257
Highest Rank Node in Block 37 is 374642
Highest Rank Node in Block 38 is 390739
Highest Rank Node in Block 39 is 396871
Highest Rank Node in Block 40 is 406300
Highest Rank Node in Block 41 is 418216
Highest Rank Node in Block 42 is 431942
Highest Rank Node in Block 43 is 437330
Highest Rank Node in Block 44 is 446565
Highest Rank Node in Block 45 is 462310
Highest Rank Node in Block 46 is 466044
Highest Rank Node in Block 47 is 481196
Highest Rank Node in Block 48 is 490478
Highest Rank Node in Block 49 is 499366
Highest Rank Node in Block 50 is 512248
Highest Rank Node in Block 51 is 514131
Highest Rank Node in Block 52 is 524510
Highest Rank Node in Block 53 is 534709
Highest Rank Node in Block 54 is 545088
Highest Rank Node in Block 55 is 555467
Highest Rank Node in Block 56 is 574139
Highest Rank Node in Block 57 is 586313
Highest Rank Node in Block 58 is 589179
Highest Rank Node in Block 59 is 605111
Highest Rank Node in Block 60 is 610392
Highest Rank Node in Block 61 is 625356
Highest Rank Node in Block 62 is 633930
Highest Rank Node in Block 63 is 640499
Highest Rank Node in Block 64 is 651680
Highest Rank Node in Block 65 is 657785
Highest Rank Node in Block 66 is 674796
Highest Rank Node in Block 67 is 678618
```

5. Extra credits:

1) Gauss-Seidel PageRank:

We use the new values as soon as they are available which can improve convergence rate.

Average number of iterations per block performed by Reducer:

Pass 0: 12
Pass 1: 6
Pass 2: 5
Pass 3: 3
Pass 4: 2
Pass 5: 1

Global Average Residual Error for each pass:
Pass 0: 3.161
Pass 1: 0.03867
Pass 2: 0.02436
Pass 3: 0.008956
Pass 4: 0.003919
Pass 5: 0.0009203

PageRank value for highest-numbered nodes in each block: (the record is node_id)
Highest Rank Node in Block 0 is 8354
Highest Rank Node in Block 1 is 16563
Highest Rank Node in Block 2 is 25567
Highest Rank Node in Block 3 is 33067
Highest Rank Node in Block 4 is 40655
Highest Rank Node in Block 5 is 50463
Highest Rank Node in Block 6 is 60843
Highest Rank Node in Block 7 is 70647
Highest Rank Node in Block 8 is 80124
Highest Rank Node in Block 9 is 94409
Highest Rank Node in Block 10 is 100862
Highest Rank Node in Block 11 is 110643
Highest Rank Node in Block 12 is 129662
Highest Rank Node in Block 13 is 139270
Highest Rank Node in Block 14 is 140574
Highest Rank Node in Block 15 is 161328
Highest Rank Node in Block 16 is 167683
Highest Rank Node in Block 17 is 174682
Highest Rank Node in Block 18 is 184001
Highest Rank Node in Block 19 is 192982
Highest Rank Node in Block 20 is 211611
Highest Rank Node in Block 21 is 222760
Highest Rank Node in Block 22 is 232579
Highest Rank Node in Block 23 is 237032
Highest Rank Node in Block 24 is 245877
Highest Rank Node in Block 25 is 258804
Highest Rank Node in Block 26 is 265528
Highest Rank Node in Block 27 is 280153
Highest Rank Node in Block 28 is 288291
Highest Rank Node in Block 29 is 294459
Highest Rank Node in Block 30 is 309302
Highest Rank Node in Block 31 is 319654
Highest Rank Node in Block 32 is 325584

```
Highest Rank Node in Block 33 is 343322
Highest Rank Node in Block 34 is 345482
Highest Rank Node in Block 35 is 355915
Highest Rank Node in Block 36 is 370257
Highest Rank Node in Block 37 is 376882
Highest Rank Node in Block 38 is 390739
Highest Rank Node in Block 39 is 396871
Highest Rank Node in Block 40 is 406300
Highest Rank Node in Block 41 is 418216
Highest Rank Node in Block 42 is 431942
Highest Rank Node in Block 43 is 440641
Highest Rank Node in Block 44 is 446565
Highest Rank Node in Block 45 is 454671
Highest Rank Node in Block 46 is 466507
Highest Rank Node in Block 47 is 480722
Highest Rank Node in Block 48 is 492387
Highest Rank Node in Block 49 is 499366
Highest Rank Node in Block 50 is 512248
Highest Rank Node in Block 51 is 514131
Highest Rank Node in Block 52 is 524510
Highest Rank Node in Block 53 is 534709
Highest Rank Node in Block 54 is 545088
Highest Rank Node in Block 55 is 563808
Highest Rank Node in Block 56 is 574139
Highest Rank Node in Block 57 is 586599
Highest Rank Node in Block 58 is 589179
Highest Rank Node in Block 59 is 599928
Highest Rank Node in Block 60 is 610392
Highest Rank Node in Block 61 is 625356
Highest Rank Node in Block 62 is 633930
Highest Rank Node in Block 63 is 640499
Highest Rank Node in Block 64 is 651680
Highest Rank Node in Block 65 is 657896
Highest Rank Node in Block 66 is 674796
Highest Rank Node in Block 67 is 678618
```

Comparison: Jacobi vs. Gauss-Seidel:

The two versions wouldn't affect the total number of MapReduce Pass too much. But their performance within the block does differ. Since Gauss-Seidel uses the new pagerank value as soon as possible, it does help the values to converge faster within the block.

2) Random Partition:

We use a hash function ***h(id) = ((id<<1) | (id+rand))%num_block*** to randomly split the nodes into different blocks where rand is a random number. We run the hash function for all the nodes, and put the partitions into the file.

We run two versions of random partition, one is 6-pass and the other is to run until converged.

1). first version:

Average number of iterations per block performed by Reducer:

Pass 0: 3
Pass 1: 3
Pass 2: 3
Pass 3: 2
Pass 4: 2
Pass 5: 2

Global Average Residual Error for each pass:

Pass 0: 2.341
Pass 1: 0.3219
Pass 2: 0.1901
Pass 3: 0.09266
Pass 4: 0.06124
Pass 5: 0.03294

2). We run 21 passes to converge.

Comparison:

1) Random Partition vs. Original Parition:

Obviously, our random partition was so bad that it need 21 passes to converge while the good partition only need 6 passes.

2) Random Partition vs. Single Node

Their performance seems similar because hash function generates poor partition. They both converge slowly.

6. How to Run our Project:

We need to preprocess the data into the format we stated in section 3. We put the new data into files (which is too large to upload in CMS).

Here is a sample:

For single input:

0 \t 1.0 \t 1,2,3
1 \t 1.0

2 \t 1.0 \t 3
3 \t 1.0 \t 1

For blocked node input:
“0+0”+”\t 1.0 \t”+”1+0,2+1,3+1”
“1+0”+”\t 1.0”
“2+1”+”\t 1.0 \t”+”3+1”
“3+1”+”\t 1.0 \t”+”1+0”

Command Line Instructions:

1) Create a job flow:

```
./elastic-mapreduce --create --alive --name "page rank" --instance-count 20
```

We will get a job flow id in the format of “j-xxx”.

2) Add a step:

a. For single nodes:

Input file: singlenodeinput
Jar: PageRank.jar
Instruction:

```
./elastic-mapreduce -j [jobflow id] \
--jar s3n://cs5300-kw427-proj2/PageRank.jar \
--main-class org.myorg.PageRank \
--arg s3n://cs5300-kw427-proj2/input/singlenodeinput  \
--arg s3n://cs5300-kw427-proj2/output_single
```

b. For blocked computation:

Jacobi:

Input file: block_out
Jacobi Jar: BlockedPageRank.jar
Instructions:

```
./elastic-mapreduce -j [jobflow id] \
--jar s3n://cs5300-kw427-proj2/BlockedPageRank.jar \
--main-class org.myorg.BlockedPageRank \
--arg s3n://cs5300-kw427-proj2/blockinput/blockinputdata \
--arg s3n://cs5300-kw427-proj2/output_blocked
```

Gauss-Seidel:

Input file: block_out
Jar: BlockedPageRank.jar
Instructions:

```
./elastic-mapreduce -j [jobflow id] \
--jar s3n://cs5300-kw427-proj2/GaussPageRank.jar \
--main-class org.myorg.GaussPageRank \
--arg s3n://cs5300-kw427-proj2/blockinput/blockinputdata \
--arg s3n://cs5300-kw427-proj2/output_Gauss
```

Random Partition:
Input file: random_block_edge
Jar: BlockedPageRank.jar
Instructions:

```
./elastic-mapreduce -j [jobflow id] \
--jar s3n://cs5300-kw427-proj2/BlockedPageRank.jar \
--main-class org.myorg.BlockedPageRank \
--arg s3n://cs5300-kw427-proj2/blockinput/random_block_edge \
--arg s3n://cs5300-kw427-proj2/output_random_blocked
```

We can also put the step into the json file in the format like:

```
{
  "Name": "Custom Jar Page Rank",
  "ActionOnFailure": "CONTINUE",
  "HadoopJarStep":
  {
    "MainClass": "org.myorg.PageRank",
    "Jar": "s3n://cs5300-kw427-proj2/PageRank.jar",
    "Args":
      [
        "s3n://cs5300-kw427-proj2/input/singlenodeinput",
        "s3n://cs5300-kw427-proj2/output_singlenode"
      ]
    }
}
```

Then execute:

```
./elastic-mapreduce -j [jobflow id] --json custom_jar_jobflow.json
```