Group : Ou Tian (ot55)   Yurun Chen (yc926) Chenxi Liu (cl969)

1.Parameters That We Use:

- Using NetID ot 55
- double fromNetID = 0.55
- double rejectMin = 0.99*fromNetID= 0.5445
- double rejectLimit = rejectMin +0.01 = 0.5545
- The number of edges: 7524968

Overall Structure of Our Code:

Simple Computation of PageRank:

```
FirstMapper:    < u , v, random number>
                      |->    < u ; v>
FirstReducer:   < u ; { v | u->v}>
                      emit   < u; P(u), { v | u->v}>
Mapper :          < u; PRt(u) , { v | u->v}>
                      |->   <u ; PRt(u) , { v | u->v}>
                      |->   <v ; PRt(u) /deg(u)| u->v>
Reducer:        <v ; PRt(v),  { w | v->w}, PRt(u) /deg(u)| u->v>
                      PRt+1(u)=(1-d)/N + d  $\sum_{u \to v}$  PRt(u) /deg(u)
                      counter+= |(PRt(u) - PRt+1(u))| / PRt+1(u)
                      emit <v ; PRt+1(u), { w | v->w} >
```

We use the Hadoop Counters to compute the residual in the same MapReduce pass that does the PageRank iteration.

Blocked Computation of PageRank:

```
FirstMapper:    < u , v, random number>
                      |->    < u ; v>
```

```
FirstReducer:   < u ; { v | u->v}>
                emit   < B(u); u , Pr(u), { B(v),  v | u->v}>
Mapper :        <B(u): u, Pr(u), {B(v), v|u->v}>
                |->    <B(u): u, Pr(u), {B(v), v|u->v}>
                |->    <B(v): v, B(u), PR(u)/deg(u)|u->v>
Reducer:        <B(v): v, Pr(u), {B(w), w|v->w}>
                 <B(v): v, B(u), PR(u)/deg(u)|u->v>
```

each Reduce task loads its entire Block into memory and does multiple in-memory PageRank iterations on the Block

```
                 emit<B(v): v, PR_new(v), {B(w), w|v->w}>
```

We use the Hadoop Counters to compute the residual in the same MapReduce pass that does the PageRank iteration. For each block reducer, we set the max iteration value as 30. And if the iteration number for convergence is less than 30, we will just iterate until convergence.

2. Results for the node-by-node computation

This is basically the sequence of average residual errors in each MapReduce pass of our implemetation:

Iteration 0 avg error 2.327161178847978

Iteration 1 avg error 0.3230273799016389

Iteration 2 avg error 0.19218709878872786

Iteration 3 avg error 0.0941505522423128

Iteration 4 avg error 0.06288266185952163

3. Results for the Blocked computation

Average residual errors average number of iterations per Block performed by our reduce tasks:

Pass 1 residual error: 2.796486683482918

Pass 1 iterations per block: 16.367647

```
Pass 2 avg residual error: 0.03811518538592881
Pass 2 avg iterations per block: 6.25
Pass 3 avg residual error: 0.023940589174437782
Pass 3 avg iterations per block: 5.0
Pass 4 avg residual error: 0.010024981571151293
Pass 4 avg iterations per block: 3.3235295
Pass 5 avg residual error: 0.004112363441472207
Pass 5 avg iterations per block: 2.1470587
Pass 6 avg residual error: 0.0015621447820439852
Pass 6 avg iterations per block: 1.4117647
Pass 7 avg residual error: 7.531557944047984E-4
Pass 7 avg iterations per block: 1.117647
```

After the 7th iteration, the PageRank values for the highest-numbered Node in each Block are listed below:

Node	PageRank
10327	1.8728504784281678E-6
20372	5.173396473900268E-7
30628	3.0634595168170754E-7
40644	2.809275717642252E-7
50461	2.996103835680399E-7
60840	2.1890460137472094E-7
70590	3.0025946728518423E-7
80117	2.1890460137472094E-7
90496	8.656791184657756E-4
100500	2.598009161845596E-7

110566	1.994696141170969E-6
120944	4.82654921606615E-7
130998	2.3420778946739548E-7
140573	6.230733969534388E-7
150952	2.1890460137472094E-7
161331	2.1890460137472094E-7
171153	5.628255833744221E-7
181513	3.508795220516106E-7
191624	5.277526511469757E-6
202003	4.0829941593633155E-6
212382	2.7250691179475626E-7
222761	0.0012126909159377432
232592	1.1363675880752636E-4
242877	5.318939922216274E-7
252937	2.1890460137472094E-7
263148	2.992874330626756E-7
273209	2.1890460137472094E-7
283472	3.367837359829545E-7
293254	2.1890460137472094E-7
303042	4.944692698445547E-6
313369	6.958228588617908E-7
323521	2.1890460137472094E-7
333882	5.035225886800546E-7
343662	2.5224609880714186E-7
353644	4.4443511822421264E-7

363928	3.8129582293826866E-7
374235	3.895731868537328E-6
384553	4.156278350076045E-7
394928	6.418100237252035E-7
404711	2.4684153137873194E-7
414616	3.115767372305049E-7
424746	2.0057776327810034E-6
434706	2.7751944425285093E-6
444488	3.829275887362308E-6
454284	4.975161090963657E-7
464397	1.105933134185094E-5
474195	9.46047127731124E-7
484049	5.916060781709682E-7
493967	2.1890460137472094E-7
503751	6.7487190742286225E-6
514130	6.199712969765474E-7
524509	9.824389683696579E-4
534708	2.514051067274733E-5
545087	0.0026089257693117465
555466	0.0011727486837300066
565845	1.5980750237226516E-6
576224	1.1045066405582565E-6
586603	9.62508522725289E-7
596584	4.029497342162977E-6
606366	4.2790440466126645E-7

616147	6.153460320150194E-7
626447	1.608733239568997E-5
636239	1.0714523726441023E-6
646021	3.1193905695897734E-7
655803	2.1890460137472094E-7
665665	9.97367072658206E-7
675447	1.1023422351373905E-6
685229	3.567398006357378E-7

4. (Optional)**Results for Gauss-Seidel**

We originally store information in reducer (nodePR, nextPR ,nodeDegree,oldPR) using hash table. To find the order of the nodes, we use a data structure TreeMap. The result for using Gauss-Seidel instead of Jacobi is showed as following:

Pass 1 residual error: 2.7972572267443048

Pass 1 iterations per block: 8.941176

Pass 2 residual error: 0.038848492835982074

Pass 2 iterations per block: 4.5588236

Pass 3 residual error: 0.025160826414488566

Pass 3 iterations per block: 3.8970587

Pass 4 residual error: 0.011110636100287495

Pass 4 iterations per block: 2.8823528

Pass 5 residual error: 0.004954586318462414

Pass 5 iterations per block: 2.1029413

Pass 6 residual error: 0.001978074186769406

Pass 6 iterations per block: 1.5441177

Pass 7 residual error: 9.051285728879354E-4

Pass 7 iterations per block: 1.2352941

5. (Optional)**Results for random partition**

Our hash function is (key*key/1000)%68.

If we also look into 7 iteration compared with Jacobi Reducer, the result for the residual and iterations each block are:

Pass 1 residual error: 2.328932945619719

Pass 1 iterations per block: 2.1176472

Pass 2 residual error: 0.3218576733476351

Pass 2 iterations per block: 2.014706

Pass 3 residual error: 0.18994747930475897

Pass 3 iterations per block: 1.7352941

Pass 4 residual error: 0.09273845370167681

Pass 4 iterations per block: 1.1470588

Pass 5 residual error: 0.061344771111889435

Pass 5 iterations per block: 1.0147059

Pass 6 residual error: 0.03301736707966668

Pass 6 iterations per block: 1.0

Pass 7 residual error: 0.026160699201727887

Pass 7 iterations per block: 1.0

Instruction: we have four Jar files, which are for node-by-node computation, blocked computation, Gauss-Seidel result and random partition result.