PageRank Project Report

Zecheng Xu (zx78) Shuo Xiong (sx78) Qianying Zhang (qz226)

We implement node-by-node and blocked PageRank algorithm, as well as Jacobi version Blocked PageRank and Random partition Blocked Pagerank

Filter Parameter:
netid: zx78
rejectMin: 0.87 * 0.99 = 0.8613
rejectLimit: 0.87 * 0.99 + 0.01 = 0.8713
number of edges selected : 7524770

We preprocessed the data (DataProcessor.java) from edges.txt and create a new file that used as input for MapReduce. The format of each line of the new file is:
< NodeID + PageRank + Degree + NeighborNodes >

All input file, output files and .jar is stored in bucket:
edu-cornell-cs-cs5300s14-zx78-project2

Simple PageRank

In simplePageRank package, there are four java classes:
1) SimplePageRank
2) SimplePageRankMapper
3) SimplePageRankReducer
4) Counter

SimplePageRank
This class contains the main method. MapReduce job is configured with certain input/output format and input/output path.
In our implementation, we stored intermediate results in HDFS to improve efficiency since it's faster than read from/write to AWS S3.

SimplePageRankMapper
Since we use the default *TextInputFormat,* the key of the InputSplit is the offset of line, value is the content of the line. In our new created input file, each line correspond to a node.
We pass two kinds of information to reducer:
1. <srcNode, "prevNodeInfo" + PageRank + srcDegree> if srcDegree = 0

<srcNode, “prevNodeInfo” + PageRank + srcDegree + dstNodes>, otherwise
this is for the purpose of reconstructing the node.
2. <dstNode, srcPageRank/srcDegree>
this is to compute new pagerank value for “dstNode”.

SimplePageRankReducer
In reducer, it simply reconstruct the node and using the second type of information passed from mapper to compute new PageRank value for this node.
Then it increments the RESIDUAL_COUNTER in Counter class. The output of reducer in current iteration is the input for mapper in next iteration.

Counter
Since we use counter, in each pass, only one MapReduce job is required. There is only one field in Counter: RESIDUAL_COUNTER.
Every reducer increments RESIDUAL_COUNTER after computing residual error.
After all reducer finishes working, we output the average residual error according to the counter value.

Result of Simple PageRank
run 6 iterations:
```
Iteration 0  Avg. residual error: 2.3385827422062664
Iteration 1  Avg. residual error: 0.3229186718255184
Iteration 2  Avg. residual error: 0.19205372241145308
Iteration 3  Avg. residual error: 0.09416565085445763
Iteration 4  Avg. residual error: 0.06287620062460779
Iteration 5  Avg. residual error: 0.03389726990207667
```

Blocked PageRank

It takes 7 passes to converge globally
There are four classes in this package, code structure is similar to simple PageRank:
1) BlockedPageRank
2) BlockedPageRankMapper
3) BlockedPageRankReducer
4) BlockCounter

BlockedPageRank
The class contains main method. Block boundaries are hard-coded for convenience.

MapReduce passes are implemented until global average residual error is lower than desired lower bound: 0.001. In the end of each termination, we output the average iteration within blocks and the average redisual error. After termination, output pageranks of highest numbered nodes in every blocks.

BlockedPageRankMapper
In mapper, we implement the *blockIDofNode* method and runs in constant time.
Three types of information were passed to reducer:

1. If srcDegree = 0:
   <srcBlock, “prevNodeInfo” + srcNode + pageRank + srcDegree>
   otherwise:
   <srcBlock, “prevNodeInfo” + srcNode + PageRank + srcDegree + dstNodes>
2. for edges within block, $\{ <u, v> \mid u \in B \wedge u \rightarrow v \}$
   <dstBlock, “BE” + srcNode(u) + dstNode(v)>
3. for edges entering$\{ <u, v, R> \mid u \notin B \wedge v \in B \wedge u \rightarrow v \wedge R = PR(u)/deg(u) \}$
   <dstBlock, “BC” + dstNode(v) + srcPR(u)/deg(u)>

BlockedPageRankReducer
We use several HashMaps to store node information and edges information.
In `IteratorBlockOnce`， we first backup current pageranks and then recompute pageranks according to edges related and then compute average residual errors within the block.
The termination condition in reducer is while the nodes within the block are converged.
After termination, increase counter REDUCER_COUNTER by 1, INNER_ITERATION_COUNTER by # of iterations of IteratorBlockOnce, RESIDUAL_COUNTER by the total residual errors in the block and corresponding PAGERANK[blockID] by the pagerank of the highest numbered nodes in block.

BlockCounter:
RESIDUAL_COUNTER: sum of residual errors
REDUCER_COUNTER: # of reducers
INNER_ITERATION_COUNTER: sum of iterations within blocks

Avg iterations = INNER_ITERATION_COUNTER / REDUCER_COUNTER

PAGERANK0 – PAGERANK67: store pagerank of highest numbered nodes in the corresponding block

Result of Blocked PageRank:

```
[pass 0]    Avg. residual error: 2.8157218713967214,    Avg. iteration: 17.485294
[pass 1]    Avg. residual error: 0.03790001912624106,    Avg. iteration: 7.161765
[pass 2]    Avg. residual error: 0.023992188521758957,    Avg. iteration: 5.867647
[pass 3]    Avg. residual error: 0.009920314455334589,    Avg. iteration: 3.897059
[pass 4]    Avg. residual error: 0.00395191477851447,    Avg. iteration: 2.558824
[pass 5]    Avg. residual error: 0.0010555445699314391,    Avg. iteration: 1.426471
[pass 6]    Avg. residual error: 6.622697747534988E-4,    Avg. iteration: 1.176471
```

sample pageranks:

```
[block 0]    Max node: 10327    PageRank: 1.87285E-6
[block 1]    Max node: 20372    PageRank: 5.2155E-7
[block 2]    Max node: 30628    PageRank: 3.0675E-7
[block 3]    Max node: 40644    PageRank: 2.8092E-7
[block 4]    Max node: 50461    PageRank: 3.0002E-7
[block 5]    Max node: 60840    PageRank: 2.189E-7
[block 6]    Max node: 70590    PageRank: 3.0011E-7
[block 7]    Max node: 80117    PageRank: 2.189E-7
[block 8]    Max node: 90496    PageRank: 8.664012E-4
[block 9]    Max node: 100500    PageRank: 2.5982E-7
[block 10]    Max node: 110566    PageRank: 2.02975E-6
[block 11]    Max node: 120944    PageRank: 4.8272E-7
[block 12]    Max node: 130998    PageRank: 2.3553E-7
[block 13]    Max node: 140573    PageRank: 6.2776E-7
[block 14]    Max node: 150952    PageRank: 2.189E-7
[block 15]    Max node: 161331    PageRank: 2.189E-7
[block 16]    Max node: 171153    PageRank: 5.6226E-7
[block 17]    Max node: 181513    PageRank: 3.4393E-7
[block 18]    Max node: 191624    PageRank: 5.27754E-6
[block 19]    Max node: 202003    PageRank: 4.14836E-6
[block 20]    Max node: 212382    PageRank: 2.6175E-7
[block 21]    Max node: 222761    PageRank: 0.0012286764
[block 22]    Max node: 232592    PageRank: 1.2014965E-4
[block 23]    Max node: 242877    PageRank: 5.6451E-7
[block 24]    Max node: 252937    PageRank: 2.189E-7
[block 25]    Max node: 263148    PageRank: 2.9941E-7
[block 26]    Max node: 273209    PageRank: 2.189E-7
[block 27]    Max node: 283472    PageRank: 3.658E-7
[block 28]    Max node: 293254    PageRank: 2.6175E-7
[block 29]    Max node: 303042    PageRank: 4.94703E-6
```

```
[block 30]   Max node: 313369   PageRank: 8.7558E-7
[block 31]   Max node: 323521   PageRank: 2.189E-7
[block 32]   Max node: 333882   PageRank: 5.1807E-7
[block 33]   Max node: 343662   PageRank: 2.5204E-7
[block 34]   Max node: 353644   PageRank: 4.3791E-7
[block 35]   Max node: 363928   PageRank: 3.5619E-7
[block 36]   Max node: 374235   PageRank: 3.26453E-6
[block 37]   Max node: 384553   PageRank: 4.272E-7
[block 38]   Max node: 394928   PageRank: 6.6669E-7
[block 39]   Max node: 404711   PageRank: 2.4565E-7
[block 40]   Max node: 414616   PageRank: 3.1166E-7
[block 41]   Max node: 424746   PageRank: 2.02404E-6
[block 42]   Max node: 434706   PageRank: 2.81297E-6
[block 43]   Max node: 444488   PageRank: 3.82919E-6
[block 44]   Max node: 454284   PageRank: 4.9751E-7
[block 45]   Max node: 464397   PageRank: 1.105941E-5
[block 46]   Max node: 474195   PageRank: 9.6575E-7
[block 47]   Max node: 484049   PageRank: 5.9304E-7
[block 48]   Max node: 493967   PageRank: 2.189E-7
[block 49]   Max node: 503751   PageRank: 6.68699E-6
[block 50]   Max node: 514130   PageRank: 6.1659E-7
[block 51]   Max node: 524509   PageRank: 9.8640168E-4
[block 52]   Max node: 534708   PageRank: 2.300763E-5
[block 53]   Max node: 545087   PageRank: 0.00257605008
[block 54]   Max node: 555466   PageRank: 0.00120386992
[block 55]   Max node: 565845   PageRank: 1.64321E-6
[block 56]   Max node: 576224   PageRank: 1.10812E-6
[block 57]   Max node: 586603   PageRank: 9.565E-7
[block 58]   Max node: 596584   PageRank: 4.10833E-6
[block 59]   Max node: 606366   PageRank: 4.2766E-7
[block 60]   Max node: 616147   PageRank: 6.2093E-7
[block 61]   Max node: 626447   PageRank: 1.565152E-5
[block 62]   Max node: 636239   PageRank: 1.02763E-6
[block 63]   Max node: 646021   PageRank: 3.1193E-7
[block 64]   Max node: 655803   PageRank: 2.189E-7
[block 65]   Max node: 665665   PageRank: 9.9855E-7
[block 66]   Max node: 675447   PageRank: 1.0857E-6
[block 67]   Max node: 685229   PageRank: 3.5609E-7
```

Randomed Block Partition

We define hash(nodeID) = nodeID
so that blockIDofNode(nodeID) = nodeID % NUM_OF_BLOCKS

It takes 22 passes to converge globally, much slower than intelligently partitioned blocks since in bad partition, most edges are between blocks and few edges are within blocks. The performance of randome block partition is basically differs little from node-by-node implementation.

Result:

```
[pass 0]    Avg. residual error: 2.3392050529342736,    Avg. iteration: 3.000000
[pass 1]    Avg. residual error: 0.322338531099608,    Avg. iteration: 2.720588
[pass 2]    Avg. residual error: 0.19120350060835012,    Avg. iteration: 2.000000
[pass 3]    Avg. residual error: 0.09352085277190784,    Avg. iteration: 2.000000
[pass 4]    Avg. residual error: 0.062049689719819,    Avg. iteration: 2.000000
[pass 5]    Avg. residual error: 0.03343268009401611,    Avg. iteration: 2.000000
[pass 6]    Avg. residual error: 0.026737168901070005,    Avg. iteration: 2.000000
[pass 7]    Avg. residual error: 0.016329769738696306,    Avg. iteration: 2.000000
[pass 8]    Avg. residual error: 0.014039672567498987,    Avg. iteration: 2.000000
[pass 9]    Avg. residual error: 0.009590845800121406,    Avg. iteration: 2.000000
[pass 10]    Avg. residual error: 0.008232902427247786,    Avg. iteration: 2.000000
[pass 11]    Avg. residual error: 0.005990730327351765,    Avg. iteration: 2.000000
[pass 12]    Avg. residual error: 0.0052024045009531105,    Avg. iteration: 2.000000
[pass 13]    Avg. residual error: 0.003905294158274054,    Avg. iteration: 2.000000
[pass 14]    Avg. residual error: 0.003378213739316799,    Avg. iteration: 2.000000
[pass 15]    Avg. residual error: 0.002622637238369467,    Avg. iteration: 2.000000
[pass 16]    Avg. residual error: 0.0022454500814906527,    Avg. iteration: 2.000000
[pass 17]    Avg. residual error: 0.0017787145639380354,    Avg. iteration: 2.000000
[pass 18]    Avg. residual error: 0.0015187079725565138,    Avg. iteration: 2.000000
[pass 19]    Avg. residual error: 0.0012179097482476395,    Avg. iteration: 2.000000
[pass 20]    Avg. residual error: 0.001034286744685084,    Avg. iteration: 1.779412
[pass 21]    Avg. residual error: 8.398437107079812E-4,    Avg. iteration: 1.000000
```

Sample pageranks:

```
[block 0]    Max node: 685168    PageRank: 3.49887E-6
[block 1]    Max node: 685169    PageRank: 7.26665E-6
[block 2]    Max node: 685170    PageRank: 7.36389E-6
[block 3]    Max node: 685171    PageRank: 7.84502E-6
[block 4]    Max node: 685172    PageRank: 3.5926E-7
[block 5]    Max node: 685173    PageRank: 4.2094E-7
```

```
[block 6]    Max node: 685174    PageRank: 6.71932E-6
[block 7]    Max node: 685175    PageRank: 8.16817E-6
[block 8]    Max node: 685176    PageRank: 2.8461E-7
[block 9]    Max node: 685177    PageRank: 6.71489E-6
[block 10]    Max node: 685178    PageRank: 6.68529E-6
[block 11]    Max node: 685179    PageRank: 6.71046E-6
[block 12]    Max node: 685180    PageRank: 5.9564E-7
[block 13]    Max node: 685181    PageRank: 3.5831E-7
[block 14]    Max node: 685182    PageRank: 5.3393E-7
[block 15]    Max node: 685183    PageRank: 4.7868E-7
[block 16]    Max node: 685184    PageRank: 4.8401E-7
[block 17]    Max node: 685185    PageRank: 3.5831E-7
[block 18]    Max node: 685186    PageRank: 6.8677E-7
[block 19]    Max node: 685187    PageRank: 3.7121E-7
[block 20]    Max node: 685188    PageRank: 4.8401E-7
[block 21]    Max node: 685189    PageRank: 4.7868E-7
[block 22]    Max node: 685190    PageRank: 4.224E-7
[block 23]    Max node: 685191    PageRank: 7.815E-7
[block 24]    Max node: 685192    PageRank: 4.8401E-7
[block 25]    Max node: 685193    PageRank: 4.8401E-7
[block 26]    Max node: 685194    PageRank: 3.5831E-7
[block 27]    Max node: 685195    PageRank: 4.7868E-7
[block 28]    Max node: 685196    PageRank: 4.5694E-7
[block 29]    Max node: 685197    PageRank: 3.2065E-7
[block 30]    Max node: 685198    PageRank: 4.8401E-7
[block 31]    Max node: 685199    PageRank: 1.45206E-6
[block 32]    Max node: 685200    PageRank: 4.6583E-7
[block 33]    Max node: 685201    PageRank: 4.6583E-7
[block 34]    Max node: 685202    PageRank: 6.5998E-7
[block 35]    Max node: 685203    PageRank: 4.2232E-7
[block 36]    Max node: 685204    PageRank: 4.8401E-7
[block 37]    Max node: 685205    PageRank: 3.5831E-7
[block 38]    Max node: 685206    PageRank: 3.7121E-7
[block 39]    Max node: 685207    PageRank: 6.8677E-7
[block 40]    Max node: 685208    PageRank: 4.8401E-7
[block 41]    Max node: 685209    PageRank: 4.8401E-7
[block 42]    Max node: 685210    PageRank: 4.8401E-7
[block 43]    Max node: 685211    PageRank: 3.5831E-7
[block 44]    Max node: 685212    PageRank: 1.79671E-6
[block 45]    Max node: 685213    PageRank: 1.79671E-6
[block 46]    Max node: 685214    PageRank: 1.79671E-6
```

```
[block 47]    Max node: 685215    PageRank: 1.79671E-6
[block 48]    Max node: 685216    PageRank: 1.79671E-6
[block 49]    Max node: 685217    PageRank: 1.79671E-6
[block 50]    Max node: 685218    PageRank: 4.7868E-7
[block 51]    Max node: 685219    PageRank: 2.189E-7
[block 52]    Max node: 685220    PageRank: 8.1196E-7
[block 53]    Max node: 685221    PageRank: 6.5998E-7
[block 54]    Max node: 685222    PageRank: 7.8008E-7
[block 55]    Max node: 685223    PageRank: 3.5831E-7
[block 56]    Max node: 685224    PageRank: 3.7482E-7
[block 57]    Max node: 685225    PageRank: 4.2232E-7
[block 58]    Max node: 685226    PageRank: 4.2232E-7
[block 59]    Max node: 685227    PageRank: 1.9606E-6
[block 60]    Max node: 685228    PageRank: 1.88119E-6
[block 61]    Max node: 685229    PageRank: 3.5831E-7
[block 62]    Max node: 685162    PageRank: 2.72815E-6
[block 63]    Max node: 685163    PageRank: 2.72815E-6
[block 64]    Max node: 685164    PageRank: 2.72372E-6
[block 65]    Max node: 685165    PageRank: 2.69502E-6
[block 66]    Max node: 685166    PageRank: 2.71929E-6
[block 67]    Max node: 685167    PageRank: 2.72815E-6
```

Gauss-Seidel

The implementation is silimar to that of Blocked PageRank, only with a little difference in the reducer.
The following equation defines the PageRank value using Gauss-Seidel method:

$$PR(i)^{(k+1)} = (1 - \alpha)/\, NUM\_OF\_NODES + \alpha \left(\sum_{i<j} PR(j)^{k} / deg(j) + \sum_{i>j} PR(j)^{(k+1)} / deg(j)\right)$$

$< j\, , i > \in$ **E**

The number of passes using Gauss-Seidel method is the same as Jacobi method, however, the average number of iterations in reducer is smaller than Jacobi.

Result:

```
[pass 0]    Avg. residual error: 2.816151834165567,    Avg. iteration: 10.088235
[pass 1]    Avg. residual error: 0.038937140946554455,    Avg. iteration: 5.102941
[pass 2]    Avg. residual error: 0.02530163499966147,    Avg. iteration: 4.367647
```

```
[pass 3]    Avg. residual error: 0.01116037507638657,    Avg. iteration: 3.220588
[pass 4]    Avg. residual error: 0.005083253142829371,    Avg. iteration: 2.367647
[pass 5]    Avg. residual error: 0.0019187138620793164,    Avg. iteration: 1.661765
[pass 6]    Avg. residual error: 8.445789655928958E-4,    Avg. iteration: 1.308824
```

Sample PageRanks:

```
[block 0]    Max node: 10327    PageRank: 1.87275E-6
[block 1]    Max node: 20372    PageRank: 5.2159E-7
[block 2]    Max node: 30628    PageRank: 3.062E-7
[block 3]    Max node: 40644    PageRank: 2.8092E-7
[block 4]    Max node: 50461    PageRank: 2.9947E-7
[block 5]    Max node: 60840    PageRank: 2.189E-7
[block 6]    Max node: 70590    PageRank: 3.0011E-7
[block 7]    Max node: 80117    PageRank: 2.189E-7
[block 8]    Max node: 90496    PageRank: 8.6470304E-4
[block 9]    Max node: 100500    PageRank: 2.5989E-7
[block 10]    Max node: 110566    PageRank: 2.03005E-6
[block 11]    Max node: 120944    PageRank: 4.7697E-7
[block 12]    Max node: 130998    PageRank: 2.3454E-7
[block 13]    Max node: 140573    PageRank: 6.3159E-7
[block 14]    Max node: 150952    PageRank: 2.189E-7
[block 15]    Max node: 161331    PageRank: 2.189E-7
[block 16]    Max node: 171153    PageRank: 5.6224E-7
[block 17]    Max node: 181513    PageRank: 3.4387E-7
[block 18]    Max node: 191624    PageRank: 5.27752E-6
[block 19]    Max node: 202003    PageRank: 4.14219E-6
[block 20]    Max node: 212382    PageRank: 2.6175E-7
[block 21]    Max node: 222761    PageRank: 0.00120921152
[block 22]    Max node: 232592    PageRank: 1.0427226E-4
[block 23]    Max node: 242877    PageRank: 5.6716E-7
[block 24]    Max node: 252937    PageRank: 2.189E-7
[block 25]    Max node: 263148    PageRank: 2.9941E-7
[block 26]    Max node: 273209    PageRank: 2.189E-7
[block 27]    Max node: 283472    PageRank: 3.657E-7
[block 28]    Max node: 293254    PageRank: 2.6175E-7
[block 29]    Max node: 303042    PageRank: 4.94705E-6
[block 30]    Max node: 313369    PageRank: 8.9347E-7
[block 31]    Max node: 323521    PageRank: 2.189E-7
[block 32]    Max node: 333882    PageRank: 5.1807E-7
[block 33]    Max node: 343662    PageRank: 2.5204E-7
[block 34]    Max node: 353644    PageRank: 4.5195E-7
```

```
[block 35]    Max node: 363928    PageRank: 3.564E-7
[block 36]    Max node: 374235    PageRank: 3.76355E-6
[block 37]    Max node: 384553    PageRank: 4.2771E-7
[block 38]    Max node: 394928    PageRank: 6.6675E-7
[block 39]    Max node: 404711    PageRank: 2.4567E-7
[block 40]    Max node: 414616    PageRank: 3.1167E-7
[block 41]    Max node: 424746    PageRank: 2.1234E-6
[block 42]    Max node: 434706    PageRank: 2.81472E-6
[block 43]    Max node: 444488    PageRank: 3.80724E-6
[block 44]    Max node: 454284    PageRank: 4.9751E-7
[block 45]    Max node: 464397    PageRank: 1.117764E-5
[block 46]    Max node: 474195    PageRank: 9.659E-7
[block 47]    Max node: 484049    PageRank: 5.9322E-7
[block 48]    Max node: 493967    PageRank: 2.189E-7
[block 49]    Max node: 503751    PageRank: 6.7826E-6
[block 50]    Max node: 514130    PageRank: 6.1659E-7
[block 51]    Max node: 524509    PageRank: 9.8739784E-4
[block 52]    Max node: 534708    PageRank: 2.281982E-5
[block 53]    Max node: 545087    PageRank: 0.00258417872
[block 54]    Max node: 555466    PageRank: 0.00120476768
[block 55]    Max node: 565845    PageRank: 1.64295E-6
[block 56]    Max node: 576224    PageRank: 1.10794E-6
[block 57]    Max node: 586603    PageRank: 9.5652E-7
[block 58]    Max node: 596584    PageRank: 4.07348E-6
[block 59]    Max node: 606366    PageRank: 4.2702E-7
[block 60]    Max node: 616147    PageRank: 6.2176E-7
[block 61]    Max node: 626447    PageRank: 1.716388E-5
[block 62]    Max node: 636239    PageRank: 1.02692E-6
[block 63]    Max node: 646021    PageRank: 3.1193E-7
[block 64]    Max node: 655803    PageRank: 2.189E-7
[block 65]    Max node: 665665    PageRank: 9.9978E-7
[block 66]    Max node: 675447    PageRank: 1.08566E-6
[block 67]    Max node: 685229    PageRank: 3.568E-7
```