MapReduce Implementation of PageRank

Alicia Chui, Jessica Kane, Eric Swidler

Getting a Subset of Edges

- randomly generated based on NetID: awc64 -> 0.46
    - rejectMin = 0.4554
    - rejectLimit = 0.465.

- we generate MapReduce input files using the files provided on cms (edges.txt, blocks.txt, nodes.txt)
    - read edges.txt line by line
    - build a list of outgoing neighbors for each node.

- Format: A file containing a single line for each node, with the node id, node's current PageRank, and list of nodes that the current node links to

Simple PageRank

One mapper, and reducers for each node

Mapper:

- **Input**: A file containing a single line for each node, with the node id, node's current PageRank, and list of nodes that the current node links to
  current node u, pr(u), {v|u->v}

- **Output**: For each node, there are two outputs.
  1) For each node that the current node links to, send the current node's page rank divided by the number of outgoing links
  2) Send the list of outgoing edges to the reducer for the current node
     node v, {{w|v->w},{PR(u)/deg(u) | u->v}}
  3) Send original page rank value for each node (used to calcualte convergence)

Simple PageRank

One mapper, and reducers for each node

Reducer:

- **Input**: A list of all outgoing links, and the PageRank value for the current node

  node v, {{w|v->w},{PR(u)/deg(u) | u->v}}

- **Compute**: PageRank of current node. d is dampening, we use 0.85

  PR(v) = (1-d) + d~(PR(u)/deg(u))

- **Emit**: A string containing the node id, the node's current PageRank, and list of nodes that the current node links to

  current node v, pr(v), {w|v->w}

Simple PageRank

- **Convergence.**

```
Pass 1 avg residual: 2.3388
Pass 2 avg residual: 0.3229
Pass 3 avg residual: 0.1919
Pass 4 avg residual: 0.0940
Pass 5 avg residual: 0.0628
```

Blocked PageRank

One mapper, reducers for each block

Mapper:

- **Input**: A file containing a single line for each node, with the node id, node's current PageRank, and list of nodes that the current node links to

  current node u, pr(u), {v|u->v}

- **Output:** For each node, there are two outputs.

  1) Block Edges: All out-edges for this node sent to node's block, contains full PR value of node.

     key: blockID(u)   value: u, pr(u), {v|u->v}

  2) Boundary Conditions: All edges to new blocks sent to receiving node's block, contains partial PR's of emitting node.

     key: blockID(v)   value: u, pr(u)/deg(u), v

Blocked PageRank

One mapper, reducers for each block

Reducer:

- **Input**: All incoming edges with PR values, list of outgoing edges

-Data structures composed for mapping Block Edges emitting node --> receiving nodes, BE receiving node --> emitting nodes, and list of Boundary Conditions edges, where in block receiving node --> outside emitting nodes.

- Iterate block to convergence (average residual < 0.001).

- **Emit**: A string containing the node id, the node's current PageRank, and list of nodes that the current node links to

current node v, pr(v), {w|v->w}

Convergence & Stats

- We compute the average residual using Hadoop Counters

- Number of passes needed for blocked PageRank convergence: 6

- Average number of iterations per Block performed by your Reduce tasks, in order of Blocks: 17.2029, 7.0725, 5.6957, 3.8551, 2.5217, 1.3188

- PageRank value for the highest-numbered Node in each Block, in order of Blocks:
1.87E-6, 5.21E-7, 3.07E-7, 2.81E-7, 3.00E-7, 2.19E-7, 3.05E-7, 2.19E-7, 8.70E-4, 2.57E-7, 2.03E-6, 4.82E-7, 2.35E-7, 6.31E-7, 2.19E-7, 2.19E-7, 5.67E-7, 3.57E-7, 5.28E-6, 4.33E-6, 2.61E-7, 0.001205, 1.07E-4, 5.29E-7, 2.19E-7, 3.00E-7, 2.19E-7, 2.95E-7, 2.61E-7, 5.16E-6, 8.18E-7, 2.19E-7, 5.17E-7, 2.49E-7, 2.27E-7, 3.54E-7, 3.19E-6, 4.14E-7, 6.80E-7, 2.46E-7, 3.12E-7, 1.80E-6, 2.80E-6, 1.06E-6, 4.98E-7, 1.11E-5, 9.49E-7, 5.94E-7, 2.19E-7, 6.71E-6, 6.17E-7, 0.001008, 2.48E-5, 0.002448, 0.001179, 1.61E-6, 1.12E-6, 9.65E-7, 4.04E-6, 4.30E-7, 6.93E-7, 1.53E-5, 9.45E-7, 3.12E-7, 2.19E-7, 1.13E-6, 1.10E-6, 3.56E-7, 3.56E-7

EC: Random Block Partition

- We programmatically generate random blocks

- Number of passes needed for convergence: 20

- Average number of iterations per Block performed by your Reduce tasks, in order of Blocks: 3.3188, 2.9275, 2.8841, 2.8406, 2.4058, 2.1159, 2.0000, 2.0145, 1.9710, 1.9710, 1.9710, 1.9710, 1.9710, 1.8696, 1.8261, 1.7681, 1.6812, 1.4203, 1.2609, 1.0290

-Lower initial average residuals (Pass 1: 2.39, Pass 2: 0.319, Pass 3: 0.18) and average block iterations (Pass 1: 3.32, Pass 2: 2.97, Pass 3: 2.88). But took much longer for overall convergence.

EC: Gauss Seidel

- Uses the new PageRank values as soon as they are available, instead of waiting an iteration

- Takes one extra overall MapRed pass, but significantly less block iterations

- **Improvement**: sort the edges by decreasing number of outbound edges in order to maximize the effect the usage of newly updated PR values

- Number of passes needed for convergence: 7
- Average number of iterations per Block performed by your Reduce tasks, in order of Blocks: Pass 1: 10.3334, Pass 2: 4.8696, Pass 3: 4.3043,
    Pass 4: 3.1014, Pass 5: 2.3333,Pass 6: 1.6377, Pass 7: 1.2174