PageRank Implementation in MapReduce

TA: Kun Li

kli@cise.ufl.edu

Hadoop version

- Your code will be tested under EMR AMI version 2.4.2
- You can develop and test your code using Hadoop 1.0.3, which is corresponding to the AWS EMR AMI version 2.4.2
- http://archive.apache.org/dist/hadoop/common/hadoop-1.0.3/
- You may have some incompatible issue using other Hadoop versions.

Development process

- First install Hadoop in your local computer and test your code with a smaller dataset.

- Then migrate to AWS EMR and test your code with the large dataset.

PageRank Driver function

```
int main(String[] args) {
  //  job 1 wiki inlink graph
   PageRank.ParseXml("wiki/data", "wiki/ranking/iter0")
  // job 2 total number of pages
   PageRank.calTotalPages("wiki/ranking/iter0", "wiki/ranking/N")
  // job 3: iterative MapReduce
   for(int run =0; run<8; run++) {
    PageRank.calPageRank("wiki/ranking/iter"+String(run),
                         "wiki/ranking/iter"+String(run+1))
   }
   // job 4: Rank page in the descending order of PageRank
   PageRank.orderRank()
  }
```

Job1: adjacency graph

- XmlInputFormat

Mahout's XMLInputFormat will process XML files and extract out the XML between two configured start / end tags. So if your XML looks like the following:

<main>
<person>
<name>Bob</name>
<dob>1970/01/01</dob>
</person>
</main>

and you've configured the start / end tags to be <person> and </person>, then your mapper will be passed the following <LongWritable, Text> pair to its map method:

LongWritable: 10 Text: "<person>\n <name>Bob</name>\n <dob>1970/01/01</dob>\n </person>"

Job1: adjacency graph

A **wikilink** (or **internal link**) links a page to another page within English Wikipedia. Links are enclosed in doubled square brackets like this:

Wikilink has 6 types of format

1. [[abc]] ---- extract “abc” out
2. [[a|b]] ---- extract “a” out
3. [[a]]b ---- extract “a” out
4. [[a]]:b ---- extract “a” out
5. ''[[a]]''b ---- extract ”a” out
6. [[a|b]]cd ---- extract ”a” out

Job1: adjacency graph

You need to exclude other types of links.

1. Interwiki links eg: [[:commons:Athens]]

2. Section linking (anchors) eg: [[#section name|displayed text]]

3. Table row linking eg: [[#top]]

4. Subpage links

Job1: adjacency graph

Extract title and wikilink in the page

<page>

<title> **AccessibleComputing**</title> -- *extract AceessibleComputing for simplicity*.

<redirect title = "Computer accessibility"> --ignore the redirect title

<text> [[Computer accessibility]]

</page>

<page>

<title> **Anarchism** </title> --extract Anarchism

<text> …. Is a [[**political philosophy**]] that advocates [[**stateless society** | stateless societies]] of defined as [[**self-goverance**|self-governed]]….

You need to write a good enough regex expressions to extract the wikilinks

Job1: adjacency graph

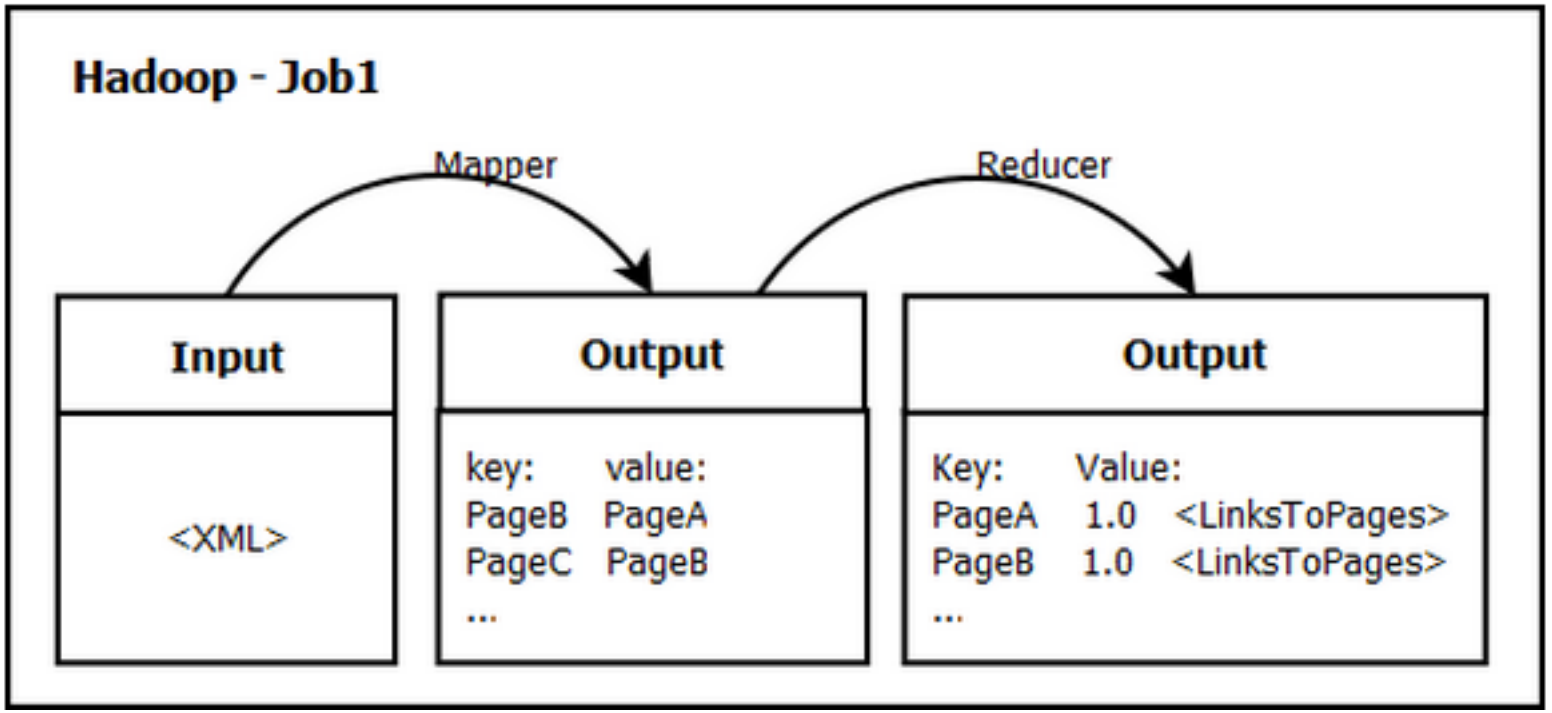
Extract wikilink

① Assume case sensitive.

② No other sophisticated processing is needed.

③ Replace empty space in title and wikilink with '_'.

Job1: adjacency graph



* The <LinksToPages> should not contain duplicate links. It also should not contain a link which points to the page itself.

Job2:PageRank Calculation

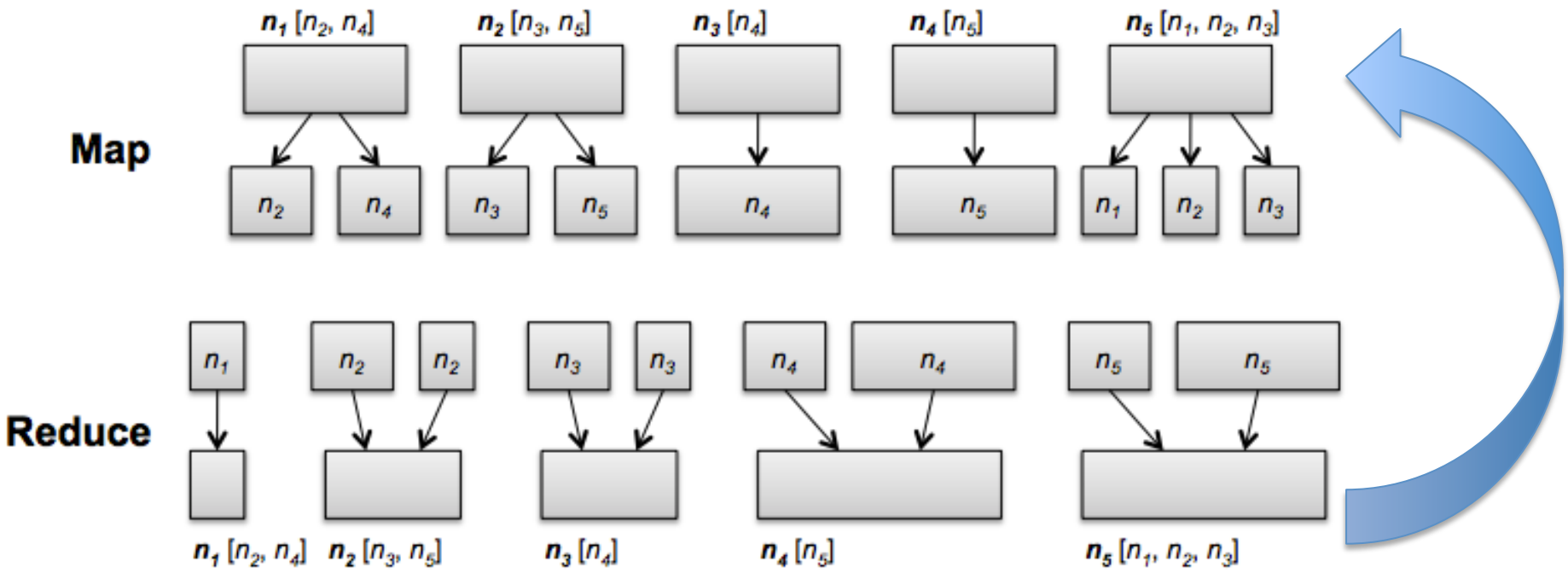
Algorithm 5.3 PageRank (simplified)

In the map phase we evenly divide up each node's PageRank mass and pass each piece along outgoing edges to neighbors. In the reduce phase PageRank contributions are summed up at each destination node. Each MapReduce job corresponds to one iteration of the algorithm. This algorithm does not handle dangling nodes and the random jump factor.

```
1: class Mapper
2:     method Map(nid n, node N)
3:         p ← N.PageRank/|N.AdjacencyList|
4:         Emit(nid n, N)                          ▷ Pass along graph structure
5:         for all nodeid m ∈ N.AdjacencyList do
6:             Emit(nid m, p)                      ▷ Pass PageRank mass to neighbors

1: class Reducer
2:     method Reduce(nid m, [p1, p2, ...])
3:         M ← ∅
4:         for all p ∈ counts [p1, p2, ...] do
5:             if IsNode(p) then
6:                 M ← p                           ▷ Recover graph structure
7:             else
8:                 s ← s + p                       ▷ Sum incoming PageRank contributions
9:         M.PageRank ← s
10:        Emit(nid m, node M)
```

Job2:PageRank Calculation



Job 3: PageRank Ordering

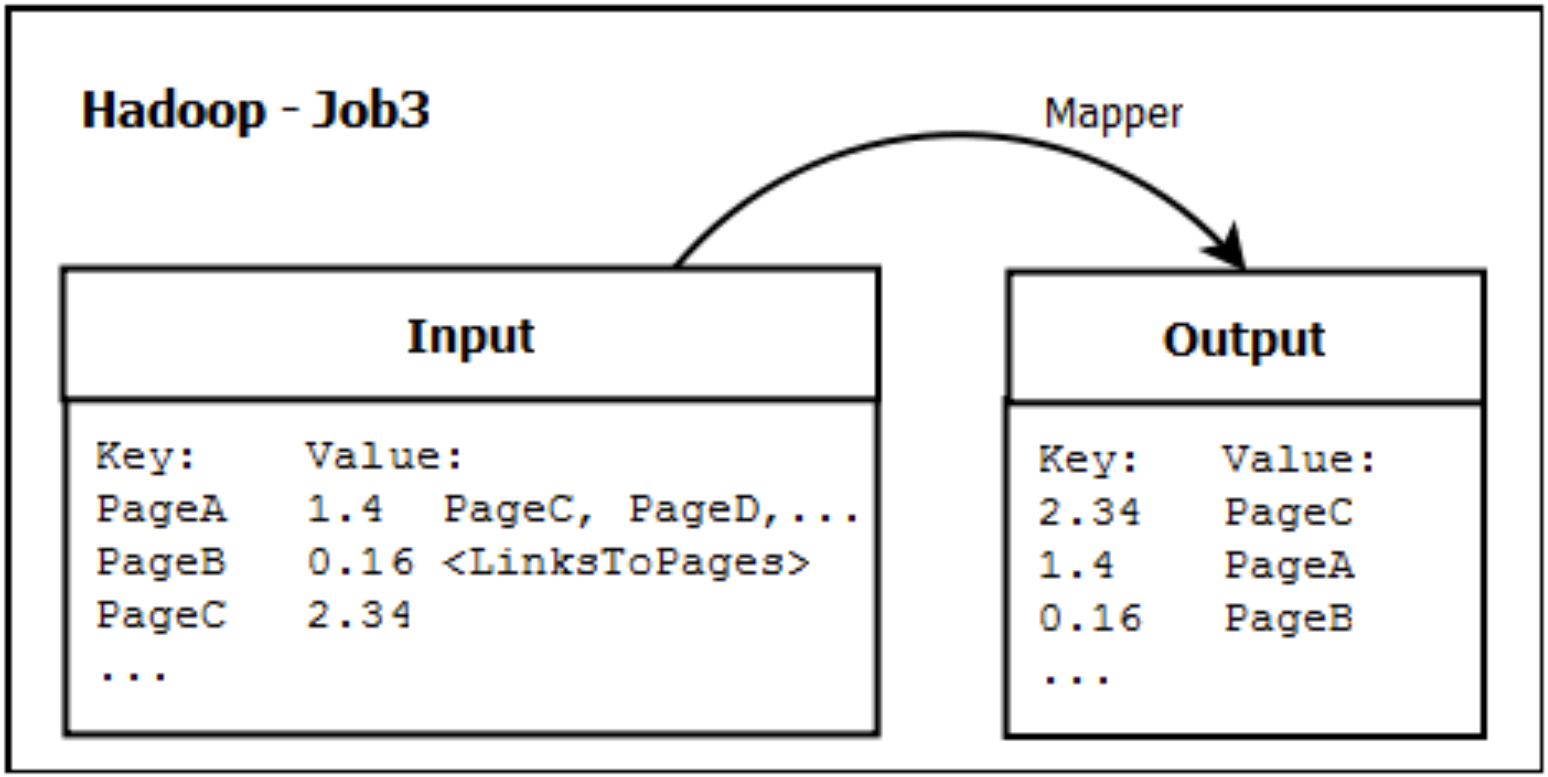
1. Filtering: only print out the page with PageRank >= 5/N (in the Map function)

2. Emit (PageRank, Page)

3. Only one Reducer

4. Output the result in the descending order of PageRank. Here you will need to override the default sorter to sort in decreasing order. *extends WritableComparator*

Job 3: PageRank Ordering



Project inputs & outputs

- Input

  spring-2014-ds/**data**/enwiki-latest-pages-articles.xml

  your-bucket-name

  results/PageRank.inlink.out

  results/PageRank.n.out

  results/PageRank.iter1.out (output file **for** iteration 1)

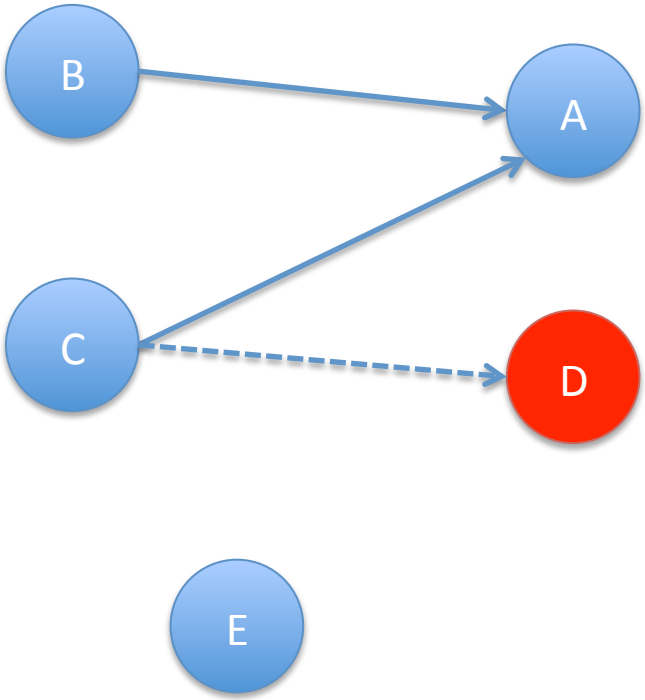
  results/PageRank.iter8.out (output file **for** iteration 8)

  logs/ (the **job** log direcotry)

  job/PageRank.jar (your **job** jar)

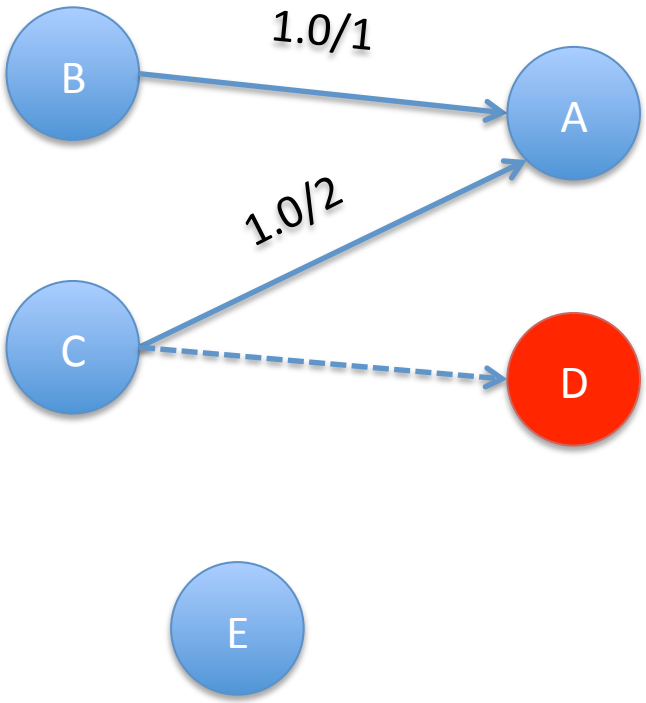
  tmp/ (temporary files, you might or might not need it)

Examples



1) Total number of pages N=4. It contains A,B,C,D
2) D is a red link. It is a non-existing page. We will not calculate the PageRank of D in our project.
3) E is a standalone page. It has no links.

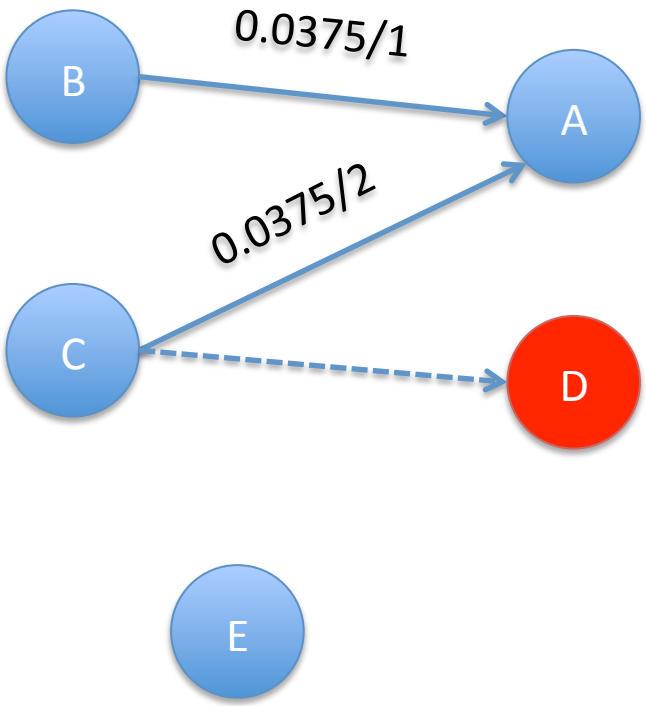
Examples: Iteration 1



1) Total number of pages N=4. It contains A,B,C,E
2) D is a red link. It is a non-existing page. We will not calculate the PageRank of D in our project.
3) E is a standalone page. It has no links. But E and its PageRank need to be printed out.

P(A) =(1-0.85)/4 + 0.85*(1.0/1 + 1.0/2) = 1.3125
P(B) = (1-0.85)/4 = 0.0375
P(C) = (1-0.85)/4 = 0.0375
P(E) = (1-0.85)/4 = 0.0375

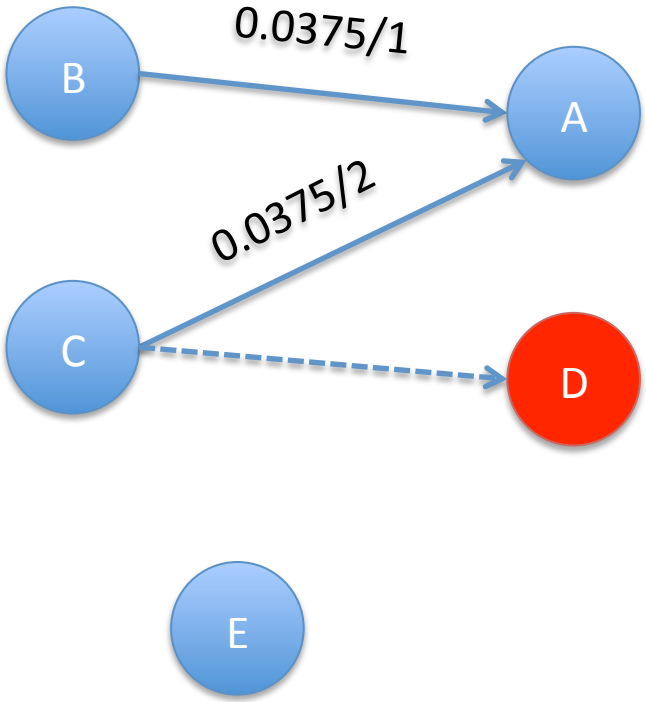
Examples: Iteration 2



1) Total number of pages N=4. It contains A,B,C,E
2) D is a red link. It is a non-existing page. We will not calculate the PageRank of D in our project.
3) E is a standalone page. It has no links. But E and its PageRank need to be printed out.

P(A) =(1-0.85)/4 + 0.85*(0.0375/1 + 0.0375/2) = 0.0853125
P(B) = (1-0.85)/4 = 0.0375
P(C) = (1-0.85)/4 = 0.0375
P(E) = (1-0.85)/4 = 0.0375

Examples: Results



PageRank.n.out

N=4

PageRank.inlink.out

Page_A

Page_B Page_A

Page_C Page_A Page_D

Page_E

PageRank.iter1.out

Page_A 1.3125

Page_B 0.0375

Page_C 0.0375

Page_E 0.0375

Questions?