

# Pagerank on Mapreduce (With Example)

Song Liu, sl9885@bristol.ac.uk

## Pagerank is a Good Thing.

It can indicate the **internal authority** of each node in a network by investigating the relationships between them.

The *original formula* is can be seen here  
<http://en.wikipedia.org/wiki/PageRank>

For simplicity, it can be written in this way:  $R_v = \beta * \sum (R_n / N_n) + (1 - \beta) / N$

Where  $R_v$  is the Pagerank of PageV and  $R_n$  represents the Pagerank of PageN which links to PageV, and  $N_n$  is the number of total links of PageN. By summing up  $(R_n / N_n)$  for all inlinks of PageV, we can get the initial Rank of PageV. To normalize this rank, we use two parameters. Beta is a parameter which is called "damping coefficient", and N is the total number of webpages.

Again, for simplicity, we don't consider the normalization: the "damping coefficient" beta and total number N.

## Let's Parallelize It

We assume the input follows the logical pattern below:

PageA -> Page1, Page2, Page3...  
PageB -> page1', Page2', Page3'...

Before the first iteration, we need to assign the initial rank for each webpage. To my experience, 0.5 is a good number for small website or network. In real calculation, this assignment can be done by a single-pass mapreduce procedure, after which, the input becomes to the following structure:

<PageA, 0.5> -> Page1, Page2, Page3...  
<PageB, 0.5> -> page1', Page2', Page3'...

This data structure indicates the outlinks for PageA, PageB... But in Pagerank, we concern more about the inlinks. So the first thing we need to do in Mapper function is to invert this data structure. Second, the factor  $R_n / N_n$  can be also easily calculated at this stage for each webpage.

```

>>>>>Pseudo Code Starts<<<<<
function Mapper
input <PageN, RankN> -> PageA, PageB, PageC...
begin
    Nn := the number of outlinks for PageN;
    for each outlink PageK
        output PageK-> <PageN, RankN/Nn>
    // output the outlinks for PageN.
    output PageN-> PageA, PageB, PageC...
end
>>>>>Pseudo Code Ends<<<<<

```

So after Mapper function, we have the following data structure:

```

PageK-> <PageN1, RankN1/Nn1>
        <PageN2, RankN2/Nn2>
        ...
        <PageAk, PageBk, PageCk...>

```

where PageN1, PageN2 is the inlinks of PageK, and PageAk, PageBk... are the outlinks. These outlinks are saved because we need them to rebuild the input format for the next iteration.

The job for Reducer function is to update the pagerank for PageK, using the ranks of inlinks.

```

>>>>>Pseudo Code Starts<<<<<
function Reducer
input PageK-> <PageN1, RankN1/Nn1>
              <PageN2, RankN2/Nn2>
              ...
              <PageAk, PageBk, PageCk...> //outlinks of PageK
begin
    RankK := 0;
    for each inlink PageNi
        RankK += RankNi/Nni * beta
    // output the PageK and its new Rank for the next iteration
    output <PageK, RankK> -> <PageAk, PageBk, PageCk...>
end
>>>>>Pseudo Code Ends<<<<<

```

It's almost done! All you need is to run this Map-Reduce procedure iteratively till convergence. To my knowledge, 15 iterations are enough in small cases, but larger case may take hundreds of iterations.

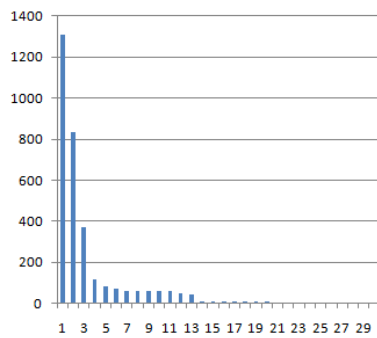
## Let's Taste It

Here is some top pageranks of today (23/Feb)'s New York Times website (I only crawled world news). There are around 14000 pages in total.

The statistics is interesting, and surely, you will find the same distribution pattern if you try to rank more website and plot more.

<http://www.nytimes.com/> 1307.420286  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html> 834.3881531  
<http://www.nytimes.com/2010/02/22/world/americas/22gays.html?bl> 372.152869  
<http://www.nytimes.com/2010/02/24/world/asia/24afghan.html?hp> 119.3507215  
<http://www.nytimes.com/2010/02/23/world/asia/23islamabad.html> 81.86431592  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?pagewanted=all> 70.64134337  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?pagewanted=2> 60.43903422  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?hpw> 59.92536076  
<http://www.nytimes.com/2010/02/23/world/asia/23afghan.html?hpw> 59.92536076  
<http://www.nytimes.com/2010/02/23/world/asia/23islamabad.html?hp> 59.92536076  
<http://www.nytimes.com/2010/02/23/world/middleeast/23mideast.html?hpw> 59.92536076  
<http://www.nytimes.com/2010/02/22/world/americas/22gays.html> 47.20198605  
<http://www.nytimes.com/2010/02/22/world/americas/22gays.html?bl=&pagewanted=all> 41.90279295  
<http://www.nytimes.com/2010/02/24/world/asia/24afghan.html> 10.56957042  
<http://www.nytimes.com/2010/02/23/world/asia/23afghan.html> 10.1774012  
<http://www.nytimes.com/2010/02/23/world/middleeast/23mideast.html> 10.02345548  
<http://www.nytimes.com/2010/02/24/world/asia/24afghan.html?hp=&pagewanted=all> 9.718268616  
<http://www.nytimes.com/2010/02/17/world/asia/17afghan.html?fta=y> 7.693364285  
<http://www.nytimes.com/2010/02/20/world/asia/20drones.html?fta=y> 7.615689791  
<http://www.nytimes.com/2002/07/07/world/afghan-official-is-assassinated-blow-to-karzai.html> 6.967736047  
<http://www.nytimes.com/2010/01/08/world/asia/08afghan.html?fta=y> 6.29221417  
<http://www.nytimes.com/2010/01/07/world/asia/07afghan.html?fta=y> 5.909407591  
<http://www.nytimes.com/2010/02/22/world/americas/22gays.html?pagewanted=all> 5.799193096  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?hpw=&pagewanted=all> 5.657414027  
<http://www.nytimes.com/2010/02/23/world/middleeast/23mideast.html?hpw=&pagewanted=all> 5.534888339  
<http://www.nytimes.com/2010/02/23/world/asia/23taliban.html?ref=asia> 5.3387006  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?pagewanted=2&hpw> 4.840375361  
<http://www.nytimes.com/2010/02/23/world/americas/23amputee.html?pagewanted=1> 4.816480315  
<http://www.nytimes.com/2010/02/16/world/asia/16intel.html> 4.671784654  
<http://www.nytimes.com/2010/02/15/world/asia/15afghan.html?fta=y> 4.292737953

Here is the rank distribution (only top 30):



## Tools

The ranks I plotted above were calculated by a small search engine which contains the Pagerank calculation on Mapreduce. Fortunately, it seems working fine with the Bluecrystal cluster.

Further detailed usage can be found in its website: <http://joycrawler.googlecode.com>

Or feel free to email me: [s19885@bristol.ac.uk](mailto:s19885@bristol.ac.uk)