# PageRank Part-II

# (Distributed)

# Nikhil Kulkarni

# VivekKumar Singh

# Of Group # 8

## Problem Statement

What was particularly important about the search engine technology was the way of ranking results of a search. Specifically, Brian and Page developed a "link analysis" algorithm called **PageRank** which ranked the pages resulting from a search according to what other and how many Web pages and sites linked to a particular Web page. Part of the ranking involved an assessment of the repute of the other Web sites/pages and assumes that more respected Web pages will link to other quality Web pages.

In this project we have implemented the PageRank algorithm in a multiprocessor architecture.

## Objectives

To implement the PageRank algorithm in a distributed architecture where the PageRank calculation will be distributed across multiple processes.

## The PageRank Algorithm

PageRank is a numeric value that represents how important a page is on the web. Google figures that when one page links to another page, it is effectively casting a vote for the other page. The more votes that are cast for a page, the more important the page must be. Also, the importance of the page that is casting the vote determines how important the vote itself is. Google calculates a page's importance from the votes cast for it. How important each vote is taken into account when a page's PageRank is calculated.

### How PageRank is calculated:

To calculate the PageRank for a page, all of its inbound links are taken into account. These are links from within the site and links from outside the site.

In the equation,

PR(A) is the PageRank of A

N is the total number of links that are connected to A

PR(ti) is the PageRank of pages ti which link to A

't1 – tn' are pages linking to page A,

'C' is the number of outbound links that a page has and

'd' is a damping factor, usually set to 0.85.

e.g. A page's PageRank = 0.15 + 0.85 \* (a "share" of the PageRank of every page that links to it)

"share" = the linking page's PageRank divided by the number of outbound links on the page

PageRank does not rank websites as a whole but is determined for each page individually. PageRank of ‘A’ is recursively defined by the PageRank of those pages which link to page A. The PageRank of pages “ti” do not influence the PageRank of page “A” uniformly. The PageRank of a page t is always weighted by the number of outbound links that the page has i.e. the more outbound links a page “t” has the less the page “A” will benefit from having a link on that page.

### Dangling Nodes:

Dangling links are simply links that point to any page with no outgoing links. They affect the model because it is not clear where their weight should be distributed, and there are a large number of them. Often these dangling links are simply pages that we have not downloaded yet.

## Implementation details

In this part of the project we implement the PageRank algorithm in a distributed environment using MPI.

The major change in the implementation of this part and the previous part of this project is that the PageRank calculation is split between different processes to make the computation faster. The processes are represented by “ranks”. The process with “rank = 0” can be called as the Root Process. All other processes can be called as the branch processes because they do the job of helping the Root process in the PageRank calculation.

The job of the two kinds of processes is as follows:

Root Process :

The job of the root process is as follows:

1. Reading the input file provided by the user.
2. Fills the data structures.
3. It sends the data structures populated from the input files and sends it to the branch processes.
4. It also broadcasts the initial PageRanks to the Branch processes.
5. It combines the PageRanks from all the branch processes calculates the final PageRank for each iteration.
6. It keeps populating the PageRanks and broadcasts it to all the processes for each iteration.
7. The root process also calculates delta which is the difference between the old and new PageRank values.

Branch Processes:

The branch processes is as follows:

1. The branch processes receive the Data structures and PageRanks which were broadcasted by the Root process.
2. The branch processes calculate the intermediate PageRank for each iteration of all the nodes assigned to it by the root process and send them back to the Root process.
3. The branch processes also calculate the dangling values if any during that iteration and send the value back to the Root process.

General flow of the implementation:

The root process populates the data structures such as the Nodes, inbounds, outbounds and the HashMap for storing the Nodes and the inbounds. It partitions the data structures for distributing among the individual processes. After partitioning the Data structures it also broadcasts the data structures required for PageRank calculation all the branch nodes according to the partitions. It also sends the delta value which is the difference between the old and new PageRank values to all the nodes.

The branch processes receive the data structures and store them for PageRank calculation. They then calculate the Page Ranks for the particular number of nodes sends them back to the root process. They also calculate the dangling values and send them to the root process.

At each iteration, the Root process receives the PageRank values and the dangling node values from the individual branch nodes and calculates the final PageRank for that iteration. 

### Data structures:

Hash Map (hash): The key of the hash map is used to store the node index as given in the input file. The value refers to a Vector that holds a collection of nodes (source) that refer the node at hand (target).

Double array (pagerank): This array holds the page ranks in double precision of all the nodes in the graph of simulated web. The index is the node number and the value is the corresponding page rank.

Integer array (outbounds): This array holds the number of outbound nodes for all the nodes in the graph of simulated web. The index is the node number and value is the number of outbound links.

Vector (bucket): This is a temporary data structure. It is used to hold the indexes (as Strings) of the source node and the target node that occur in each line of the input file.

Vector (inbounds): This data structure is used to hold the indexes of all the nodes that have an incoming link to the node in consideration. This is used along with the hash data structure.

Integer (COUNTER): this data field is used to hold the number of nodes in the simulated web.

## Performance Analysis

Above given graphs show the runtime performance with respect to time of our implementation of the PageRank algorithm.

## Technology used:

The Distributed PageRank algorithm was implemented in Java using the Eclipse IDE.

MPI was used to create a distributed environment, where processes can interact with each other using formal MPI routines and methods. For implementing MPI in Java, MPJ was used which is the Java MPI library.

## Conclusions :

In this part we have successfully scaled the sequential version of the PageRank algorithm to incorporate the features of distributed computing. We have successfully established the PageRank calculation process using multiple processes.

## Acknowledgements:

We are thankful to Prof. Judy Qui, Stephen Wu and Ikhyun Park for their guidance and support.

## References:

1. PageRank wiki: <http://en.wikipedia.org/wiki/PageRank>
2. Workshop.net: <http://www.webworkshop.net/pagerank.html>
3. MPJ : http://mpj-express.org/