# Implementing Page Rank Algorithm using MPI

**Purshottam Vishwakarma**
School of Informatics and Computing
Indiana University, Bloomington
pvishwak@indiana.edu

**Bitan Saha**
School of Informatics and Computing
Indiana University, Bloomington
bsaha@indiana.edu

## Introduction:

Page Rank Algorithm, developed by Larry Page and Sergey Brin at Stanford University as part of their research, is the brain behind Google search engine. It uses probability distribution across all the pages over the web and assigns a value between 0 and 1 to each web page which determines the popularity of the page. Computation of page rank is an iterative process and the accuracy of the page rank is directly proportional to the number of iterations performed. Considering billions of web pages over the internet it is a highly compute intensive task and requires the algorithm to be run in parallel over a distributed environment. This Project implements the sequential version of the algorithm in a far smaller scale i.e. we would rank only 1000 web pages. *We have tried to make this algorithm highly efficient (3x faster) as compared to our peers.*

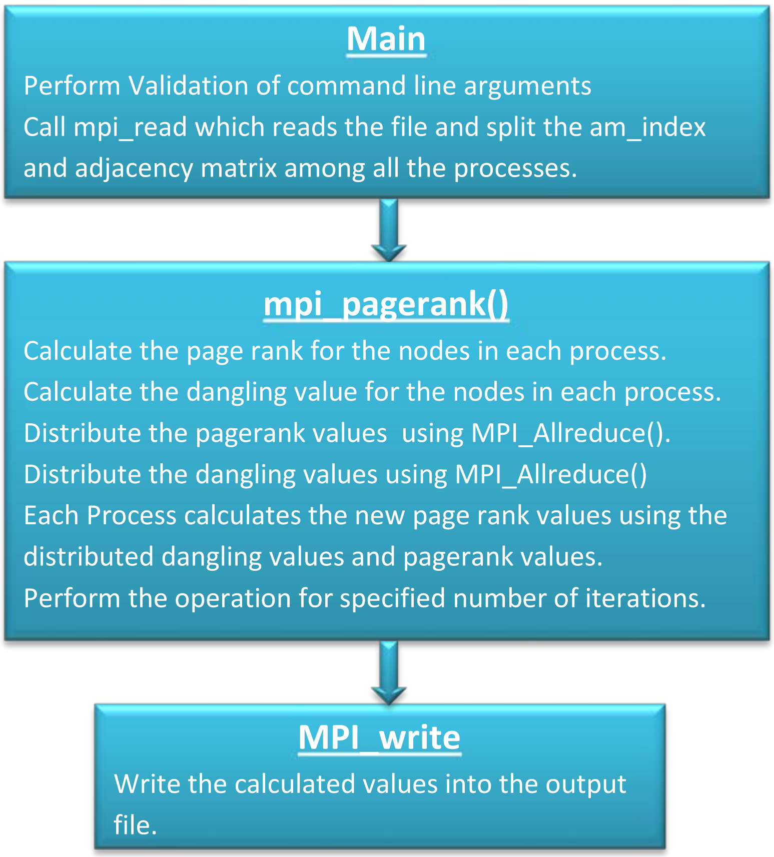
## Theory:

In the general case, the PageRank value for any page *i* can be expressed as:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

PR, pagerank (a probability value)
pi , a page under consideration
L(pi), the number of outbound links on page pj
d, damping factor which can be set between 0 and 1 (It is usually set d to 0.85)
N, total number of pages.

## Architecture design (e.g. architecture flowchart and text)



**Main**

Perform Validation of command line arguments
Call mpi_read which reads the file and split the am_index and adjacency matrix among all the processes.

**mpi_pagerank()**

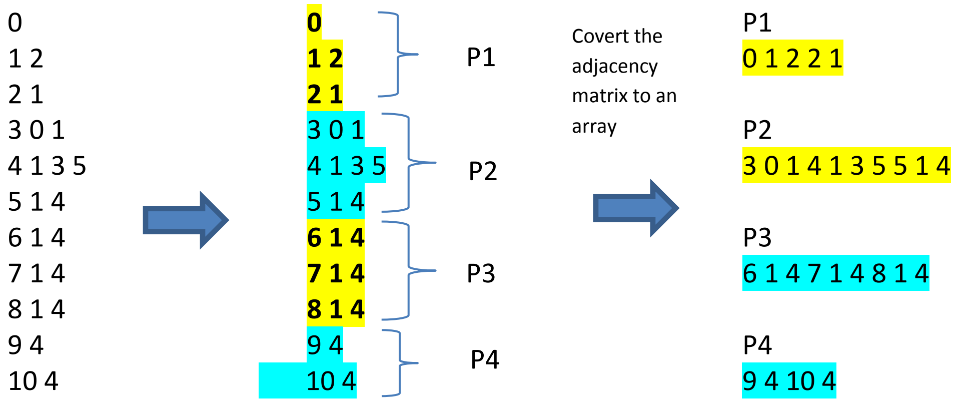
Calculate the page rank for the nodes in each process.
Calculate the dangling value for the nodes in each process.
Distribute the pagerank values using MPI_Allreduce().
Distribute the dangling values using MPI_Allreduce()
Each Process calculates the new page rank values using the distributed dangling values and pagerank values.
Perform the operation for specified number of iterations.

**MPI_write**

Write the calculated values into the output file.

## Implementation:

- The algorithm splits the entire adjacency matrix among the number of processes specified.
- For example, the below adjacency matrix is split among 4 processes in the following manner.



0
1 2
2 1
3 0 1
4 1 3 5
5 1 4
6 1 4
7 1 4
8 1 4
9 4
10 4

P1: **0** / **1 2** / **2 1**
P2: 3 0 1 / 4 1 3 5 / 5 1 4
P3: **6 1 4** / **7 1 4** / **8 1 4**
P4: 9 4 / 10 4

Covert the adjacency matrix to an array

P1
0 1 2 2 1

P2
3 0 1 4 1 3 5 5 1 4

P3
6 1 4 7 1 4 8 1 4

P4
9 4 10 4

- Each process keeps the track of the root node (1st column in the original adjacency matrix) using am_index data structure.
- Each process calculates the associate the page rank of the outbound nodes and dangling values of the root node. After calculating them each process sends these two values

(pagerank_values_table and the dangling values) to MPI and it sums up the values received from all the processes and then sends back the aggregated values to all the processes. Thus each process now has the aggregated values of the pagerank.

- Similarly MPI_Allreduce() receives individual process dangling values, sums them up and sends back the aggregate values back to each process.
- Each process then adds the damping factor into the calculated value to get the final value of page rank.
- The above steps are performed for the specified number of iterations.

Below is the diagrammatic representation of the algorithm for **iteration 1**.

| **Process 1** | **Process 2** | **Process 3** | **Process 4** |
|---|---|---|---|
| [0] = 0.000000<br>[1] = 0.090909<br>[2] = 0.090909<br>[3] = 0.000000<br>[4] = 0.000000<br>[5] = 0.000000<br>[6] = 0.000000<br>[7] = 0.000000<br>[8] = 0.000000<br>[9] = 0.000000<br>[10] = 0.000000 | [0] = 0.045455<br>[1] = 0.121212<br>[2] = 0.000000<br>[3] = 0.030303<br>[4] = 0.045455<br>[5] = 0.030303<br>[6] = 0.000000<br>[7] = 0.000000<br>[8] = 0.000000<br>[9] = 0.000000<br>[10] = 0.000000 | [0] = 0.000000<br>[1] = 0.136364<br>[2] = 0.000000<br>[3] = 0.000000<br>[4] = 0.136364<br>[5] = 0.000000<br>[6] = 0.000000<br>[7] = 0.000000<br>[8] = 0.000000<br>[9] = 0.000000<br>[10] = 0.000000 | [0] = 0.000000<br>[1] = 0.000000<br>[2] = 0.000000<br>[3] = 0.000000<br>[4] = 0.181818<br>[5] = 0.000000<br>[6] = 0.000000<br>[7] = 0.000000<br>[8] = 0.000000<br>[9] = 0.000000<br>[10] = 0.000000 |
| Dangling value = 0.090909 | Dangling value = 0 | Dangling value = 0 | Dangling value = 0 |

**MPI_Allreduce**

| |
|---|
| [0] = 0.045455<br>[1] = 0.348485<br>[2] = 0.090909<br>[3] = 0.030303<br>[4] = 0.363636<br>[5] = 0.030303<br>[6] = 0.000000<br>[7] = 0.000000<br>[8] = 0.000000<br>[9] = 0.000000<br>[10] = 0.000000 |
| Sum_Dangling = 0.090909 |

## **Performance Analysis**:

We have benchmarked our program with the sequential version of the program and found that the MPI version of page rank works faster than the sequential version. Below are the timings for the sequential version and the MPI version with 1000 nodes, 10 processes and 10 iterations.

| | |
|---|---|
| Sequential Version of PageRank | 1 s |
| MPI Version of PageRank (5 processes) | 1 ms |
| MPI Version of PageRank (10 processes) | 3 ms |
| MPI Version of PageRank (15 processes) | 4 ms |
| MPI Version of PageRank (20 processes) | 8 ms |

## Results:

Below are the page ranks of the top 10 ranking URL numbers.

```
4      0.134060
34     0.125075
0      0.111998
20     0.084226
146    0.066848
2      0.048978
12     0.021937
14     0.017361
16     0.012771
66     0.012066
```

## Acknowledgements:

We acknowledge the efforts of our Instructor **Judy Qiu** and Associate Instructors **Ikhyun Park & and Tak Lon Wu** in making us understand the basic concepts behind page rank.

## References

1. Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, Stanford University, WWW7 Proceedings of the seventh international conference on World Wide Web 7 , 1998
2. http://en.wikipedia.org/wiki/Markov_chain
3. http://en.wikipedia.org/wiki/Adjacency_matrix
4. http://en.wikipedia.org/wiki/PageRank
5. http://www.open-mpi.org/doc/v1.4/