Web Mining in Cloud Computing Framework: RANKING

Project Report submitted in partial fulfillment of the requirement for the degree of

Master of Technology

in

Computer Science & Engineering

under the supervision of

Suman Saha

By

Ashish Rawat **(132207)**

Jaypee University Of Information Technology
Waknaghat, Solan - 173234, HP, INDIA

May 2015

Certificate

This is to certify that project report entitled "Web Mining in Cloud Computing Framework: RANKING" , submitted by Ashish Rawat in partial fulfillment for the award of degree of Master of Technology in Computer Science and Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Suman Saha
Assistant Professor

Acknowledgements

This project could not have been at the stage it is right now had it not been for the cooperation of **Mr. Suman Saha**, our project guide, who was always there to tell me how to go about my project in a systematic manner and who always took out time to help me with our technical and non-technical doubts at various stages of the project also he kept faith in my ability to complete the project well and on time. Last but not the least; it was our fellow students who came to our rescue whenever we got stuck in any piece of code or otherwise.

Date: **Ashish Rawat**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

The dimension of World Wide Web (The Internet) is in billions in terms of web pages and increasing rapidly. With the diversity of web pages available on the web, the high degree relevant information retrieval becomes a major issue. Such huge number of pages not only make the computation complex but also raises the issues of fault tolerance and time complexity. Computing ranking for such large number of web graph on a particular system, makes it prone to system failure and time taking. The present work proposes a distributed ranking system to attain fault tolerance and speedy calculation of Pagerank vector. The computation of rank vector is performed by implementing Pagerank on Mapreduce framework. The pagerank vector is calculated via spectral analysis to make the computation even faster and the results are compared to traditional pagerank scores.

CHAPTER 1

INTRODUCTION

The information retrieval regarding any certain query, on the web, has several replies which create challenges in a relevant information excavation. The most relevant information excavation has become a vital issue among several billions of web pages existing with the presence of malicious pages and programs to manipulate the search result of search engines. There exist various ranking algorithms using various approaches to rank the web pages, e.g.: PageRank, Hits, Opic Algorithm, Ranking hubs and Authorities using Matrix Function etc. Among these algorithms Pagerank algorithm has an advantage over others as it not only checks the link from pages but also checks from which page it is coming from. In addition it calculates single quality measure i.e., the pagerank vector. A detail description of pagerank algorithm will be discussed in section ahead. Beyond such advantages of pagerank, it takes underrepresented amount of time for the computation of pagerank vector. Computation of such complex algorithm on a single system makes it single system failure prone system. So the time and single system failure are the two problems we are addressing in this project. This work is fundamentally based on computing pagerank on MapReduce frame work. MapReduce is a widespread distributed programming model for processing complex and huge size data. Map reduce was incorporated in the year of 2004, aiming web index, by Google. Basically it is a distributed programming model and an scalable implementation for processing and generation of huge data sets.

The Pagerank computation is done by using the concept of spectral analysis. The maximum eigenvalue is calculated using spectral analysis. In accordance to spectral analysis the eigenvalue λ of a adjacency matrix M can be determined by the following relation [2]

$$\lambda_1 = \max_Z \frac{Z^T M Z}{Z^T Z}$$

Proof: Suppose M be a symmetric matrix with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq .... \geq \lambda_n$, and the corresponding orthogonal eigenvectors $v_1, v_2, ..., v_n$

Let, $Z = c_1v_1 + c_2v_2...c_nv_n$,be the characteristic equation for matrix A

Now

$$Z^TMZ = (c_1v_1 + c_2v_2...c_nv_n)^TM(c_1v_1 + c_2v_2...c_nv_n)$$

$$= (c_1v_1 + c_2v_2...c_nv_n)^T((c_1\lambda_1v_1 + c_2\lambda_2v_2...c_n\lambda_nv_n))$$

Since $v_1, v_2, ..., v_n$ are orthogonal.

$$= \sum_{i=1}^{n} c_i^2\lambda_i$$

Similarly ,

$$Z^TZ = (c_1v_1 + c_2v_2...c_nv_n)^T(c_1v_1 + c_2v_2...c_nv_n) = \sum_{i=1}^{n} c_i^2$$

So,

$$\frac{Z^TMZ}{Z^TZ} = \frac{\sum_{i=1}^{n} c_i^2\lambda_i}{\sum_{i=1}^{n} c_i^2} \leq \lambda_1 \frac{\sum_{i=1}^{n} c_i^2}{\sum_{i=1}^{n} c_i^2}$$

$$\frac{Z^TMZ}{Z^TZ} = \lambda_1$$

i.e., the maximum eigenvalue.

Note:

The whole scenario of this presentation is based on mining on the web. Let us understand the term Web Mining first.

1.1 Web Mining

Web Mining is a large discipline and one of its form is Ranking, which is very productive and useful form of information retrieval. It is the branch of Data Mining that automatically discovers or extracts the information from web documents. We can difine it as drawing out the interesting, implicit information and potentially useful patterns from activity related to the World.

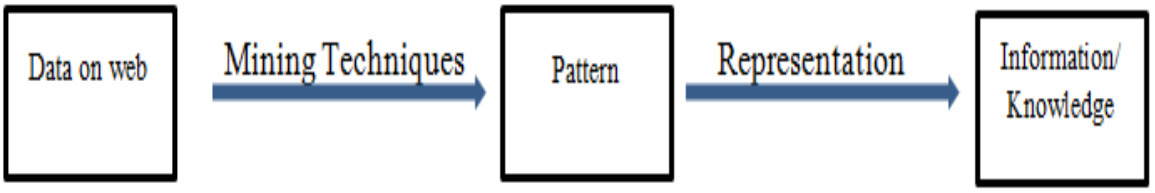

FIGURE 1.1: Web Mining Process

Web Mining Process

1. **Resource finding:** It includes the task of retrieving concerned web documents.
2. **Information selection and pre-processing:** It Includes the automatic selection and pre-processing of specific information from retrieved web Resources.
3. **Generalization:** It automatically discovers general Patterns across all the information exist on web.
4. **Analysis:** It involves the validation and interpretation of the mined patterns[3].

Note:

After understanding the Process Web Mining. As per the title of my Dissertation "Web Mining in Cloud Computing Framework" lets now understand the concept of MapReduce.
Reason: MapReduce is scalable, reliable computing model which optimizes the system performance but is hardly used in ranking mechanism.

1.2 MapReduce

MapReduce is a general-purpose parallel programming plateform for complex and huge size data computing.Map reduce was incorporated in the year of 2004, aiming web index, by Google.
MapReduce is a distributed programming model and an accompanying implementation for operating and generation of huge data sets.

1. A **map** function that performs a function with a key/value pair to generate a set of intermediate key/value pairs.
2. A **reduce** function that collects the intermediate results by various map functions and computes on the result provided by map function [4].

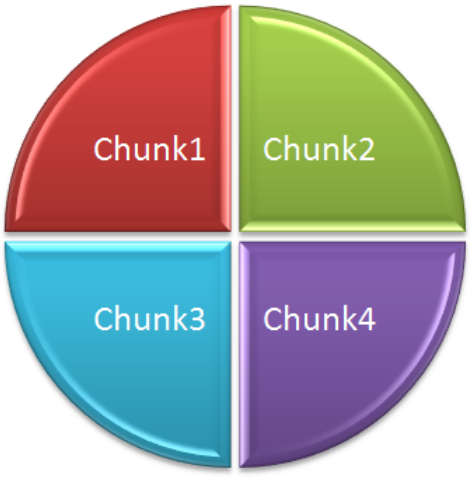

FIGURE 1.2: Data Partitioning

Prior to Map reduce phase prepossessing of data is required. Data is to be partitioned into equal size chunks. Chunks of data need to be partitioned into equal size, as intermediate results are further merged by *reducer* process.

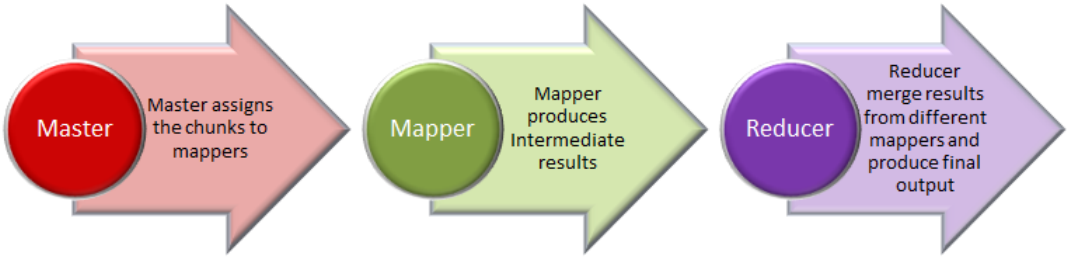

FIGURE 1.3: Map Reduce Process

As you can see in the Figure 1.3 first of all *Master* process assigns the chunks to various *Mapper* process. Now every *Mapper* process will read the the assigned chunks and produce the intermediate results. Now every *Mapper* process will send their intermediate results to *reducer* process. In next phase the *reducer* process will sort and merge the intermediate results dent by various *mapper* process. Finally *reducer* process will apply the user defined function to produce final result.

1.2.1 Execution phase of a MapReduce Application

1. Master function of MapReduce partitioned files/data into number of pieces and copies called chunks.

2. Master picks a jobless worker i.e. *mapper* and allocates a map task.

3. The *mapper* function analyze the key/value pairs of the input chunk(data) and produces the intermediate results.

4. The *mapper* sends the output key/value pair in the memory. Now forwards these intermediate key/value pair to the master and then master further forwards it to the *reducer*.

5. After collecting the intermediate results, the *reducer* sort and merge them .

6. Now for every intermediate results, the *reduce* function apply the corresponding processing.

Note:

After detail information of both Web mining and MapReudce, we need to understand how the ranking of web pages is achieved.

1.2.2 Fault Tolerance

Map reduce is designed to deal with huge size data. Various machine/process working in parallel must be fault tolerant to deal against system failure. To handle with the issue of fault tolerance the *Master* process pings every *mapper* process in certain interval. If no response is generated from any *mapper* process and it is considered dead and its task has been assigned to some other idle *mapper* process. Now *reducer* process has been notified about the change i.e. re computation of particular task so that in case non responding *mapper* sends their result, it will be discarded.

1.3 Ranking in Web

Web ranking signifies the rank of web pages with respect to the quality if information they comprises. The quality of information contained, can be jugged by the endorsement that other pages provide to certain other pages via hyperlink.

When the user fires a query, the index is consulted to get the documents most relevant to the query. The relevant documents are then ranked according to their degree of relevance, importance etc.

Notations: Web gives us a unique way to measure the importance of a document. With the web, documents are connected using hyperlinks. If a page B has a link to page A, then it says page A has a backlink from page B. We can view backlinks as a type of endorsement. The more backlinks a page have, the more important the page is.

FIGURE 1.4: Web ranking [1]

Note:

Before we discuss the various best algorithms in ranking we need to understand the concept of Web Graph, where these algorithms are actually going to be implemented.

1.4 Web Graph

- Web is a collection of hyperlinked documents. We can view the web as a directed graph where each page is a node and a hyperlink from one page to another is captured by a directed edge.
- If page A has a link to page B, then there should be a directed edge from node A to node B. Every page has a number of forward edges (out edges) and backlinks (in edges). Page A has a backlink from page B if there is a hyperlink from page A to page B.
- Backlink can be viewed as a type of endorsement and the count of backlinks is regarded as a measure of importance of a page.
- This idea was used earlier in the field of citation analysis. But deciding the importance of a page based on backlink count pose another problem in terms

of link farms. Link farms are a group of web pages that link to each other. In this way the vicious web page owner can have a high backlink count by setting up another n number of web pages each having a hyperlink to that page.

- There exist various algorithms that solves this problem by not only counting the backlinks, but also noting the page from which the link is coming from. These algorithms are discussed in ahead section Literature survey.

1.5 Problem Statement

The relevant information retrieval on the web create challenges with thousands of web pages increasing every day and problem becomes more severe with a variety of web pages exist on the web. The most relevant information retrieval becomes a major issue at present with the existence of billions of web pages along with malicious web pages to manipulate the search result. Such large number of pages takes unprecedented amount of time in calculation if rank of web pages.

On the context of problem discussed above there is a need to devise a well defined algorithm which not only facilitate users with most appropriate and relevant replies of their fired query but also compute rank in fewer timer.

1.6 Motivation

- The size of the World Wide Web (The Internet) is 3.2 billion web pages[5] and increasing rapidly. The information retrieval creates new challenges on web every day.
- With the diversity of web pages exist on the web. The most relevant information retrieval becomes a major issue among billions of web pages specially in the presence of a malicious creator of a web pages and programs to manipulate the search result of search engines.
- This arises the need of a Ranking algorithm which delivers the most relevant information and do it in timely manner.

1.7 Objective

- A Distributed Ranking system which returns the most relevant information about the query.
- The calculation of Pagerank of web graph on Mapreduce framework by splitting web graph into equal size of chunks. Now varies mapper will be allocated one or more than one chunks to compute the pagerank and particular mapper will send the result to reducer. In last reducer will sort the result from various mapper and compute the pagerank vector of the given web graph.

CHAPTER 2

PRELIMINARIES AND BACKGROUND

Most cited and Popular Ranking algorithms

1. HITS Algorithm [Classical][6].
2. PageRank algorithm [Classical][7].
3. PageRank without hyperlinks[8].
4. Ranking hubs and Authorities using Matrix Function[9].
5. OPIC Algorithm[10].
6. Full-text and Topic Based Author Rank and Enhanced Publication Ranking[11].
7. and so on.

2.1 Hyperlink Induced Topic Search

- Hyperlink Induced Topic Search, in short HITS algorithm is a link analysis algorithm developed by Jon Kleinberg [6].
- This algorithm computes two values for any page:-
  1. **Hub Score:** The value of its links to other pages.

$$\vec{h}(v) = \sum_{v \to y} a(y)$$

  2. **Authority Score:**The value of the page's content.

$$\vec{a}(v) = \sum_{y \to v} h(y)$$

2.1.1 Topic Specific Search

HITS algorithm is a topic specific search. At prior a subset of web pages containing based on query keywords is created. This is carried off by first firing the query and determining an initial set of documents feasible to the query (say 100 documents) . This set is termed as root set for the particular query. The subset of these pages is formed by appending every page that is either link to any of the pages in the root set or is linked by some page contend the root set. The idea behind this procedure is as follows. A good authoritative page in the fired query perhaps not be made up of text of the fired query (for example, home page of Apple does not consist of the term computer). So, it might not be fetched as a query result. But the good hub pages are supposed to contain the query words only. If we able to acquire good hub pages, we concatenate the good authoritative pages by going through the "out-going links" of the particular page.

2.1.2 Execution phase

The HITS algorithm works as follows:

- It determines the root set of the compatible documents by launching the query at first.

- Then the base set of hubs and authorities from the root set is created.
- After that it determines the values of AA^T and A^TA , from the adjacency matrix of the base set of documents (subset of web-graph).
- It then computes the eigenvalues and eigenvectors.
- Finally, it outputs the top scoring hubs and authorities.

Note: With the advancement of time lots of changes have been made in HITS algorithm. So the latest version of HITS algorithm is follows:

Algorithm 1 HITS

Require: Web Graph.

Ensure: Top hubs and authorities.

1. Analyze the web graph.
2. Computes the values of AA^T and A^TA from the adjacency matrix of the web-graph.
3. Computes the value of $\vec{h}$ and $\vec{a}$ by computing the values eigenvectors of AA^T and A^TA respectively.

- The adjacency matrix AA^T and A^TA are computed from given matrix A.
- Now the value of $\vec{h}$ and $\vec{a}$ are computed as the values eigenvectors of AA^T and A^TA respectively
- Finally, the top scoring hubs and authorities are examined, where authority vector represents the rank vector.

2.1.3 Example

The Hits can be explained using the following example. Assume a collection of five documents 1, 2, 3, 4 & 5. Here is the example of a graph of 5 nodes created in R. Directed graph representing the hyperlinks from one page to another. As you can see in the Figure 2.1 an arrow heading from node 1 to node 3, representing the edge from node 1 to node 3.

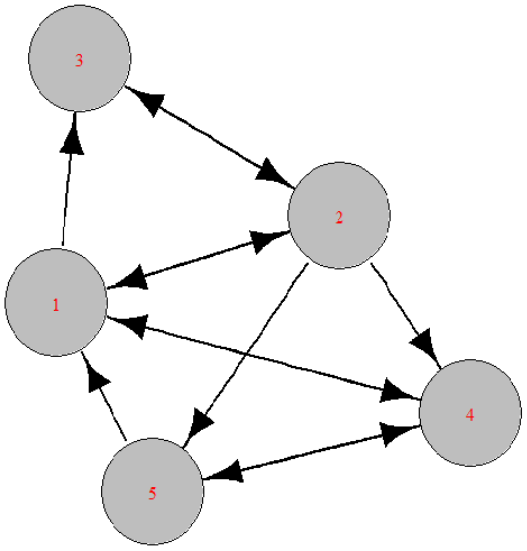

FIGURE 2.1: HITS example

Now converting the above graph into transition matrix. The transition matrix of size 5×5 is.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Initializing the hub vector $\vec{h}$ by unity.

$$\vec{h} = \begin{pmatrix} .2 \\ .2 \\ .2 \\ .2 \\ .2 \end{pmatrix}$$

Calculating normalized authority vector, $\vec{a} = A^t\vec{h}$

$$\vec{a} = \begin{pmatrix} .25 \\ .16 \\ .16 \\ .25 \\ .16 \end{pmatrix}$$

Calculating normalized hub vector, $\vec{h} = A\vec{a}$

$$\vec{h} = \begin{pmatrix} .23 \\ .33 \\ .06 \\ .16 \\ .20 \end{pmatrix}$$

Second Iteration

$$\vec{a} = \begin{pmatrix} .24 \\ .10 \\ .20 \\ .27 \\ .17 \end{pmatrix}$$

$$\vec{h} = \begin{pmatrix} .22 \\ .35 \\ .04 \\ .16 \\ .20 \end{pmatrix}$$

Third iteration.

$$\vec{a} = \begin{pmatrix} .25 \\ .09 \\ .20 \\ .27 \\ .18 \end{pmatrix}$$

$$\vec{h} = \begin{pmatrix} .22 \\ .35 \\ .03 \\ .17 \\ .20 \end{pmatrix}$$

Forth iteration.

$$\vec{a} = \begin{pmatrix} .25 \\ .09 \\ .20 \\ .27 \\ .18 \end{pmatrix}$$

$$\vec{h} = \begin{pmatrix} .22 \\ .35 \\ .03 \\ .17 \\ .20 \end{pmatrix}$$

Since the state of equilibrium is achieved, so we can come up to the conclusion that,

$$\vec{a} = \begin{pmatrix} .25 \\ .09 \\ .20 \\ .27 \\ .18 \end{pmatrix}$$

is the rank vector for the graph shown in Figure 2.1

Index	Hub	Authority	Rank
1	0.22	0.25	2
2	0.35	0.09	5
3	0.03	0.20	3
4	0.17	0.27	1
5	0.20	0.18	4

TABLE 2.1: Hub and Authority vector

The table 2.1 representing Hub , Authority Vectors and Rank of nodes. The rank of the nodes are decided by there authority score. Node with high authority score is ranked up.

2.1.4 Simulation Results

A directed graph of 100 nodes have been used to simulate HITS algorithm. Directed edges of the graph represents the hyperlink from one node to another. The graph created in R is as follows:

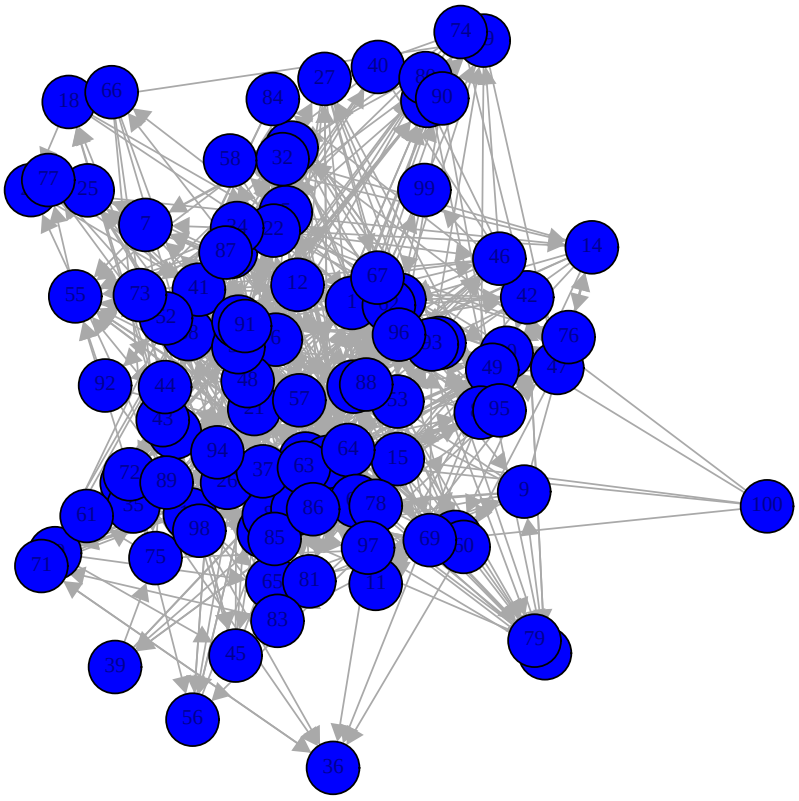

FIGURE 2.2: Directed graph with 100 nodes

Index	Hub	Authority
1	0.073	0.084
2	0.080	0.045
3	0.106	0.081
4	0.053	0.027
5	0.038	0.048
6	0.020	0.015
7	0.056	0.111
:	:	:
:	:	:
98	0.155	0.061
99	0.077	0.040
100	0.08	0.01

TABLE 2.2: Hub and Authority vector

These are Hubs and Authority score of all the nodes of graph where authority score represents rank of the node. Node with high authority score is ranked up.

2.2 PageRank

PageRank is a link evaluation process that determines the importance of a document by analyzing the link structure of a hyperlinked set of documents and its a way of measuring the importance of a website. It is developed by Larry Page (co-founder of Google).

The idea behind the Page Rank is the importance of the web page that can be determined by glancing at the pages linked to it [7]. The simple backlink count was sufficient for well controlled document collection of citation analysis. But in web, there is hardly any control. Millions of pages can be automatically created and linked with each other to manipulate the backlink count [2]. As web consists of conflicting profit making ventures, any evaluation strategy which counts replication features of web pages is bound to be manipulated. PageRank extends the idea of backlink by "not evaluating links from every page equally, and by normalizing by the count of links on a page." [7].

2.2.1 Damping Factor

The Page rank algorithm states that an imaginary surfer who is randomly going on links, may stop surfing after some time. The probability, at any step, that the person will continue to surf is a damping factor $\frac{1-d}{N}$ (N is the total count of pages). The constant d is a damping factor which can be set between 0 and 1. We have taken d as 0.85.

2.2.2 Random Surfer

The Page rank algorithm follows the model of random surfer who after following several links and stride to any other page at random. The value Page Rank of a certain page represents the possibility that the random surfer will move on to a page by clicking on the link. If a page has no outbound links, it becomes the sink and abolish the random surfing spree. If random surfer get onto a sink page, it randomly picks another URL and carry on surfing again. While pages with no outbound links are assumed to link each and every page in the collection as stated by pagerank algorithm. The Page rank score are therefore distributed evenly to rest of the pages.

Algorithm 2 Pagerank

Require: $G = (V, E)$

Ensure: Rank(V)

1. Given a graph G, where V is the vector and E is the edges in adjacency Matrix.
2. Analysis of $G = (V, E)$
3. Determine transition matrix of the given graph G.
4. PageRank vector,
   $PR(u) = \frac{1-d}{N} + d\sum_{v\in<u,v>} \frac{PR(v)}{L(v)}$

- $PR(u)$ denotes the pagerank of node u ,which is the summation of pagerank $PR(v)$ of all v nodes pointing to it, where $L(v)$ are the number of outgoing links from node v [12].

- Here N is the number of nodes in the web graph and parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85 [7].

- In terms of matrices, PageRank values are the entries of eigenvector R, Let say $u_1, u_2, u_3....u_n$ are the various web pages in the graph so R is the solution of equation [10].

$$R = \begin{bmatrix} PR_{u1} \\ PR_{u2} \\ PR_{u3} \\ \vdots \\ PR_{un} \end{bmatrix}$$

- Exploding the matrix.

$$R_n = \begin{bmatrix} \frac{1-d}{N} \\ \frac{1-d}{N} \\ \vdots \\ \frac{1-d}{N} \end{bmatrix} + d \begin{bmatrix} f(u_1,u_1) & f(u_1,u_2) & \cdots & f(u_1,u_n) \\ f(u_2,u_1) & f(u_2,u_2) & \cdots & f(u_2,u_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(u_n,u_1) & f(u_n,u_2) & \cdots & f(u_n,u_n) \end{bmatrix} R_{n-1}$$

  where $f(u_i, u_j)$ indicates the presence of a link from page u_i to u_i.

- If there's no link present in between,

$$f(u_i, u_j) = 0$$

- Else it is normalized so that for each j ,

$$\sum f(u_i, u_j) = 1.$$

2.2.3 Example

The PageRank can be explained using the following example. Assume a collection of four documents 1, 2, 3 & 4. Since PageRank is the probability of arriving at a document, the initial probability for each of the document is equal at 1.

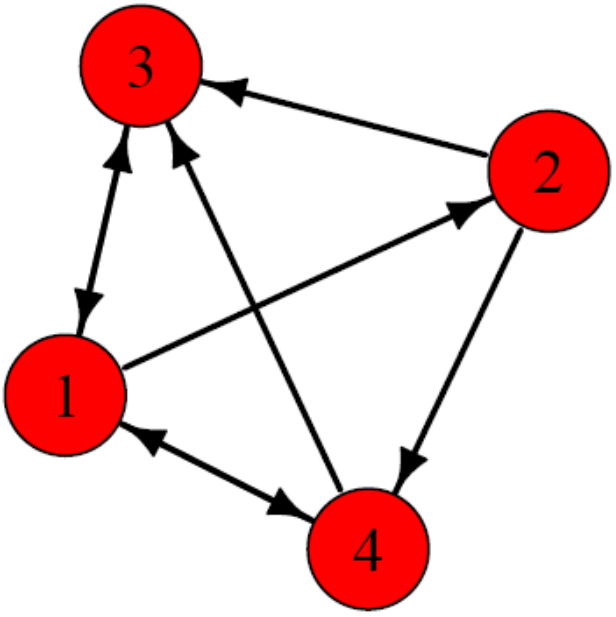

FIGURE 2.3: PageRank example

In this example, since every page should transfer its weightage to all the outgoing links i.e. pages. So generally a with initial pagerank value 1 and x outgoing links, will give endorsement of $\frac{1}{x}$ to every x page [13]. The transition matrix for the above example will be.

$$A = \begin{bmatrix} 0 & 1 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

Lets initialize the rank vector v by value .25 for every page i.e. all sum to 1. Every incoming link adds up the importance of a web page, so we upgrade the

rank of each page by increasing the current value with the endorsement value of the incoming links. Or we can simply multiply the matrix A with v . Now At first step the new importance vector is $v_1 = Av$. The process will now follow iterative steps, thus at second step the upgraded importance vector will be $v_2 = A(Av) = A^2v$. Further computations is as follows:
Initializing rank vector,

$$v = \begin{pmatrix} .25 \\ .25 \\ .25 \\ .25 \end{pmatrix}$$

First iteration:

$$Av = \begin{pmatrix} .37 \\ .08 \\ .33 \\ .20 \end{pmatrix}$$

Second iteration:

$$A^2v = \begin{pmatrix} .43 \\ .12 \\ .28 \\ .19 \end{pmatrix}$$

Third iteration:

$$A^3v = \begin{pmatrix} .35 \\ .14 \\ .29 \\ .20 \end{pmatrix}$$

Forth iteration:

$$A^4v = \begin{pmatrix} .39 \\ .11 \\ .29 \\ .19 \end{pmatrix}$$

Fifth iteration:

$$A^5 v = \begin{pmatrix} .39 \\ .13 \\ .28 \\ .19 \end{pmatrix}$$

Sixth iteration:

$$A^6 v = \begin{pmatrix} .38 \\ .14 \\ .29 \\ .19 \end{pmatrix}$$

Seventh iteration:

$$A^7 v = \begin{pmatrix} .38 \\ .12 \\ .29 \\ .19 \end{pmatrix}$$

Eighth iteration:

$$A^8 v = \begin{pmatrix} .38 \\ .12 \\ .39 \\ .19 \end{pmatrix}$$

Since after few steps equilibrium is achieved, so we can come up to the conclusion that,

$$A^7 v = \begin{pmatrix} .38 \\ .12 \\ .29 \\ .19 \end{pmatrix}$$

is the rank vector for the graph shown in Figure 2.2

Index	**Pagerank Score**	**Rank**
1	0.38	1
2	0.12	4
3	0.29	2
4	0.19	3

TABLE 2.3: Pagerank vector

The table 2.2 representing Pagerank score and Rank of nodes. The rank of the nodes are decided by there Pagerank score. Node with high Pagerank score is ranked up.

Major differences between HITS and Page Rank algorithm

- HITS algorithm calculates two quality measures on every document i.e., hub and authority, as opposed to a single score in case of PageRank.
- HITS is not commonly used by search engines.
- PageRank is more robust against spam web pages as it is not easy for a web page owner to get links from highly ranked web pages.

Note:

The PageRank algorithm favors the older pages than recent ones. As older pages are expected to have more number of citations from the important pages than a page just introduced. Hence, page rank should not be used as a standalone metric. It should be used parameter only.

2.2.4 Simulation Results

A directed graph of 100 nodes have been used to simulate page rank algorithm. Directed edges of the graph represents the hyperlink from one node to another. The graph created in R is as follows:

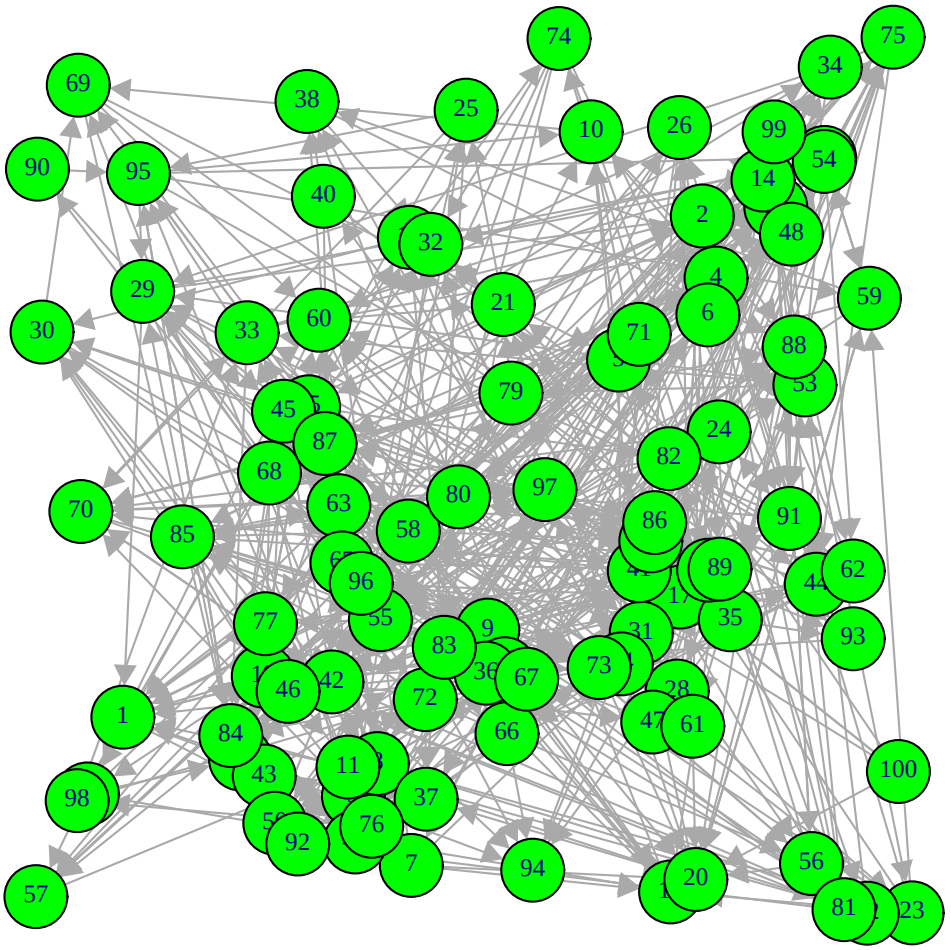

FIGURE 2.4: Directed graph with 100 nodes

The implementation of pagerank algorithm is done again in R language. The implementation results of pagerank algorithm are taken as ground truth to compare the results of proposed approach.

Index	**PageRank**
1	0.0097
2	0.0086
3	0.0089
4	0.0120
5	0.0065
6	0.0076
7	0.0098
:	:
:	:
98	0.0088
99	0.0089
100	0.0015

TABLE 2.4: PageRank vector

Table 2.4 depicts the pagerank score of above example, in Figure 2.4. The page/node with highest pagerank score ranks first among all.

2.3 Online Page Importance Calculation

- Online Page Importance Calculation (OPIC) is a link analysis algorithm that can compute the page importance at the time of crawling i.e., digitally online and with finite resource [10].
- This algorithm maintains two values for each page: cash and history. All pages are given some initial cash. When the page is crawled, it distributes its current cash equally to all pages it points to.
- The cash of a node records the sum of the cash obtained from the page since the last time it was crawled.
- The history variable stores the sum of the cash acquired from the page since the go-off of the algorithm till the last time it was crawled.
- The importance of a page is given by the value in the history of the page. The idea is that the flux of cash at a page is proportional to its importance [10].

2.4 Ranking hubs and Authorities using Matrix Function

- The notions of sub-graph centrality and connectivity, of the adjacency matrix of the given graph, have been used with efficaciously in the analysis of undirected networks[9].
- An expanse of these effective measures to directed networks is incorporated for ranking web pages i.e,. hubs and authorities.
- The comprehension of bipartization is applied in this work.
- The bipartite graph algorithm for determining hubs and authorities is compared to the traditional HITS algorithm .

Algorithm 3 Ranking Hubs and Authorities using Matrix Function

Require: Web Graph.

Ensure: Bipartite graph with In and Out Degrees.

1. Analyze the web graph.
2. Convert the Web graph into Bipartite graph .
3. Computes the quality measure, In-degree In and Out-degree Out.
4. Juxtaposition the results with HITS algorithm.

- The In-degree and Out-degree represents the Hubs and Authority score respectively, where authority score represents rank of the node. Node with highest In-degree score is ranked up among all.

- The notions of sub-graph centrality and communicability, of the adjacency matrix of the underlying graph, have been effectively used in the analysis of undirected networks[9].
- An extension of these measures to directed networks is incorporated for ranking web pages i.e,. hubs and authorities.
- The concept of bipartization is used in this algorithm.
- The bipartite graph method for computing hubs and authorities is compared to the well known HITS algorithm [14].

2.4.1 Example

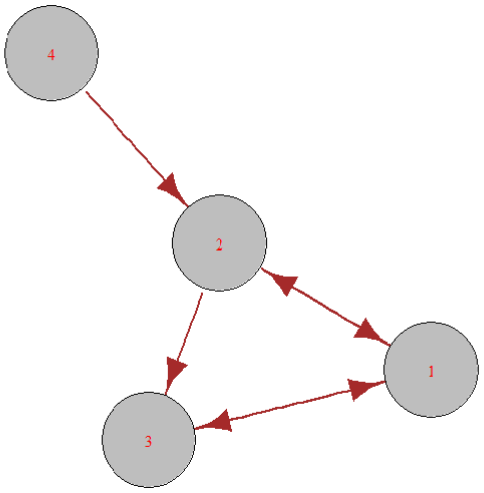

FIGURE 2.5: Example of four nodes

Considering one example of an small directed network, and its matrix A

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Now converting above example into bipartite graph. A bipartite graph is a graph which divides its vertices into two sets. The two sets are partite sets such that every vertex of first set contains an edge to atleast one vertex in second set. Now the outgoing edges of first set represents the outdegree of vertices while the incoming edges in second set represents the indegree of the vertices. The bipartite graph of Figure 2.5 is shown in Figure 2.6

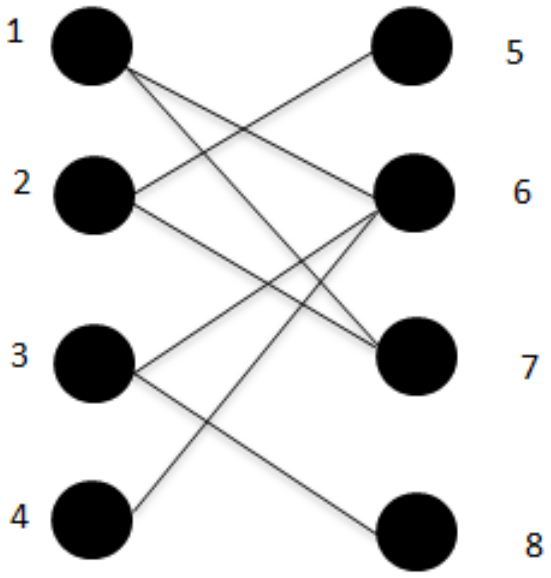

FIGURE 2.6: The bipartite network with adjacency matrix A

The corresponding bipartite graph as shown in previous figure, the hubs and authorities scores are computed simply using in-degree and out-degree counts and the result is as follows:

Node	Out-degree	In-degree
1	2	1
2	2	3
3	2	2
4	1	1

TABLE 2.5: In-degree and out-degree counts of the bipartite graph

Note: These are In-degree and Out-degree which represents the Hubs and Authority score respectively, of every node in graph where authority score represents rank of the node. Node with higher In-degree score is ranked up.

2.4.2 Simulation Results

A directed graph of 10 nodes have been used to simulate "Ranking Hubs and Authorities using Matrix function". Directed edges of the graph represents the hyperlink from one node to another. The graph created in R Studio is as follows:

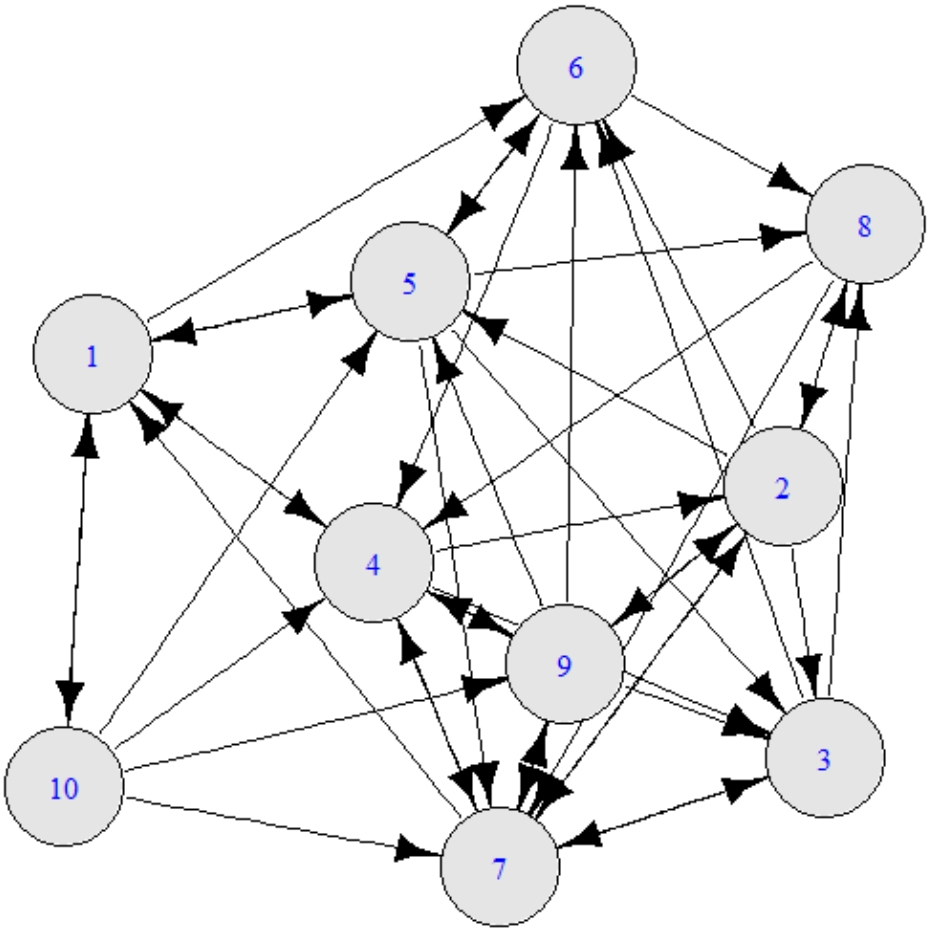

FIGURE 2.7: Directed graph with 10 nodes

As you can see in the Figure 2.8 the adjacency matrix of 10×10 is created in R studio. Here function $AM(v, e)$ is created to build the adjacency matrix which take vertices v and edges e as input while discarding the self loop.

```
> v <- 10
> e <- 50
> m1 <- AM(v,e)
> m1
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    1    1    1    0    0    0     1
 [2,]    0    0    1    0    1    1    1    1    1     0
 [3,]    0    0    0    0    0    1    1    1    0     0
 [4,]    1    1    1    0    0    0    1    0    1     0
 [5,]    1    0    1    0    0    1    1    1    0     0
 [6,]    0    0    0    1    1    0    0    1    0     0
 [7,]    1    1    1    1    0    0    0    0    1     0
 [8,]    0    1    0    1    0    0    1    0    0     0
 [9,]    0    1    1    1    1    1    1    0    0     0
[10,]    1    0    0    1    1    0    1    0    1     0
> 
```

FIGURE 2.8: Snapshot of matrix created in R

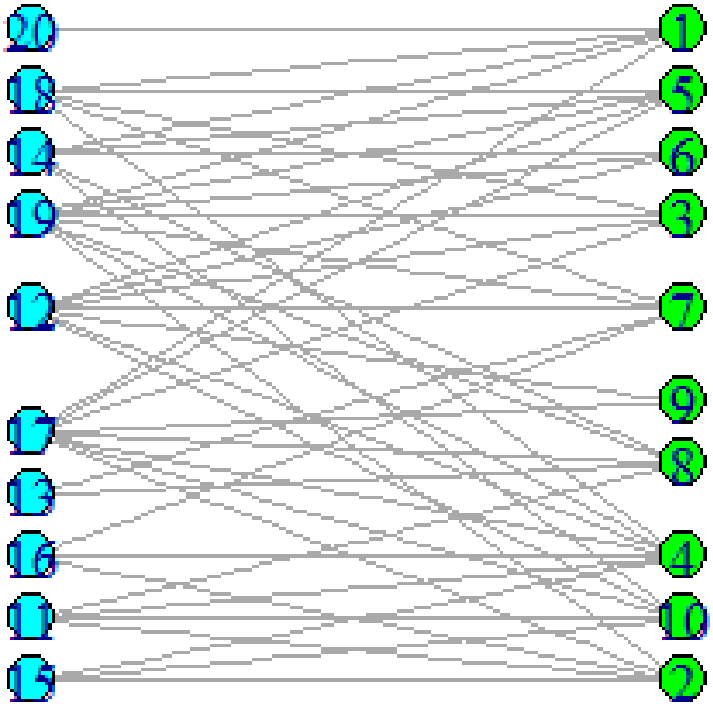

FIGURE 2.9: Bipartite graph with 10 nodes

Next step is to create the Bipartite graph graph of above example, Figure 2.9 . Bipartite graph as shown in Figure 2.8, again created in R studio. The number of

outgoing edges vertices blue in color is the outdegree of vertices while the incoming edges on green color vertices represents the indegree of the vertices.

Index	Indegree	Outdegree
1	3	4
2	5	6
3	7	5
4	6	4
5	4	6
6	3	6
7	5	3
8	6	2
9	5	5
10	1	3

TABLE 2.6: In-degree and out-degree counts of the bipartite graph

Table 2.4 represents the In-degree and Out-degree which further represents the Hubs and Authority score, respectively, of every node in graph. The authority score represents rank score of the node. Node with higher In-degree score is ranked up.

CHAPTER 3

PROPOSED APPROACH

3.1 Proposed Approach

The proposed work is two fold, first is pagerank computation on mapreduce framework and other one is using the concept of spectral analysis to compute pagerank again on mapreduce framework. Both these approaches goal at distributed, fault tolerant and time efficient ranking system.

3.1.1 Pagerank computation on mapreduce framework

In context to the problem of fault tolerance and time of computation we proposed the distributed ranking system. The framework of map reduce has been used here. This approach deals with Map reduce implementation of pagerank algorithm. The prim motive is to reduce time of computation, make the system fault tolerant while no information loss. The implementation of this approach is done in R language. The algorithm has been tasted over varying size of directed graph and found to be working well for every graph. The Proposed Approach can be formalized in algorithmic form as follows:

Algorithm 4 Mapreduce pagerank algorithm

Require: $G = (V, E)$

Ensure: Rank(V)

1. Given a graph G, where V is the vector and E is the edges in adjacency Matrix.
2. Analysis of $G = (V, E)$
3. Partition of G in S_n chunks.
4. $map(S_1, PR_1)$
   return $list(S_1, PR_1)$
5. $reduce(S_1, list(PR_1))$
   return $Rank(V)$

Description: Consider a directed graph $G = (V, E)$ such that, V is the vertices and E is the edges in adjacency Matrix of web pages. After the analysis of the matrix and it is divided into n chunks say $S_1, S_2, ...S_n$ of equal sizes. Now these chunks assigned to mapper function. Mapper function will calculate the Pagerank Matrix equation PR. After that various mapppers will send their intermediate results to *reducer*. Now *reducer* will sort the intermediate results and calculate the $Rank(V)$ vector.

The algorithm starts with analysis of directed web graph. Analyses phase starts with counting number of vertices and the edges present in the directed web graph. Analyses concludes that the matrix of web graph is sparse, where most of the entries are 0. The sparse matrix generated by directed graph is converted into transition matrix form. The transition matrix states links present between two pages/nodes. Now existence of an directed edge from page u_i to page v_j the (u_i, v_j) entry will be 1 else 0 in the matrix. The self loops are renounced at the time of analysis phase i.e., the value (u_i, u_i) for every i^{th} is 0 or we can say that the diagonal of the transition matrix is 0.

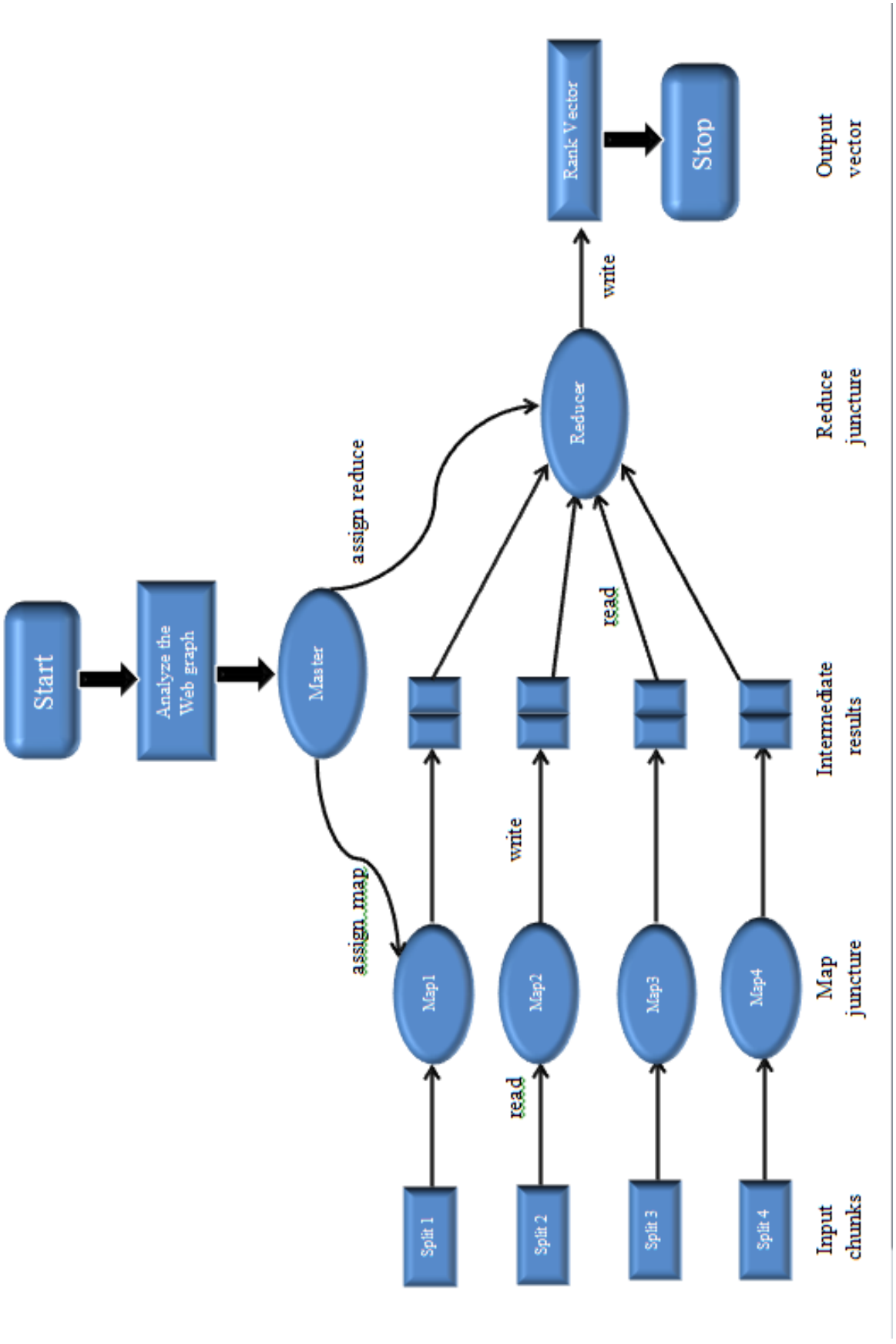

FIGURE 3.1: Pagerank computation on Map Reduce framework.

After analyses phase the transition matrix is divided into equal size chucks as depicted in Figure 3.1. The partitioning is done to distribute the load among different machines. Now system will jump to map phase where master function will assigned the particular task to mappers. Each mapper will read the chunk assigned to it and computes the matrix pagerank equation for the same. Individual mappers will now process on their assigned chunks and generate their results which are termed as intermediate results in our system. The matrix pagerank equation Rn as described above in $algorithm2$ [7].

$$R_n = \begin{bmatrix} \frac{1-d}{N} \\ \frac{1-d}{N} \\ \vdots \\ \frac{1-d}{N} \end{bmatrix} + d \begin{bmatrix} f(u_1,u_1) & f(u_1,u_2) & \cdots & f(u_1,u_n) \\ f(u_2,u_1) & f(u_2,u_2) & \cdots & f(u_2,u_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(u_n,u_1) & f(u_n,u_2) & \cdots & f(u_n,u_n) \end{bmatrix} R_{n-1}$$

Every intermediate result will be sent to the reducer function. Once intermediate result of every partitioned chunk are received by reducer, it will sort and merge them all. Now the eigenvalueλ and corresponding to this eigenvalue, eigenvector R of merged result be computed by reduce function. The equation used for computing eigenvalue and eigenvector is:

$$(M - \lambda I)R = 0$$

,where M is the transition matrix and I is the identity matrix.This eigenvector R represents the rank vector of the input directed graph. The page with top eigenvalue in corresponding eigenvector will be ranked top most among all the pages.

Using above relation to compute eigenvalue and eigenvector gives the absolute results but with the complexity of $O(n^3)$ per step in practice. This takes huge time to determine the rank vector of the transition matrix and the problem becomes more severe when matrix size reaches in millions. To overcome this problem we used concept of spectral analysis to compute the highest eigenvalue of the transition matrix.

3.1.2 Processes

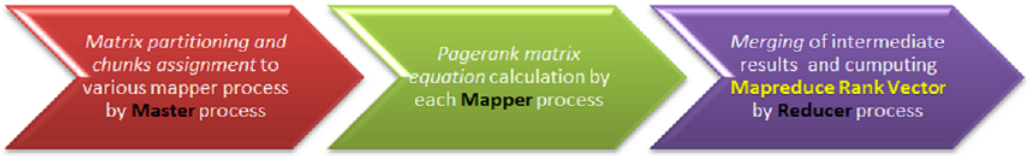

FIGURE 3.2: Process flow in Pagerank computation on mapreduce framework

Three different processes comprises the whole system. The algorithm starts with *Master* process which assigns task to various *Mapper* process and in last *Reducer* process will feed intermediate results from every *mapper* to produce final output. The functioning of these processes in this algorithm are as follows:

1. *Master* process: The transition matrix of the given directed will be the input of *master* process. The transition matrix will be *partitioned* into equal size chunks here and each of these chunks will be *assigned* to an idle *mapper*.

2. *Mapper* process: Assigned chunks will be processed here by *mapper* process to compute the *Pagerank matrix equation* which will be termed as intermediate results. The intermediate results will now be forwarded to *reducer* process.

3. *Reducer* process: All the intermediate results i.e. *Pagerank matrix equation* from various *mappers* will now be *merged* together. Once the whole intermediate results are merged, the user defined function to calculate the *eigenvalue* of the regenerated matrix will be applied. Corresponding to largest *eigenvalue* the *eigenvector* will be computed. This *eigenvector* is the output of the algorithm i.e. **Mapreduce Rank Vector**.

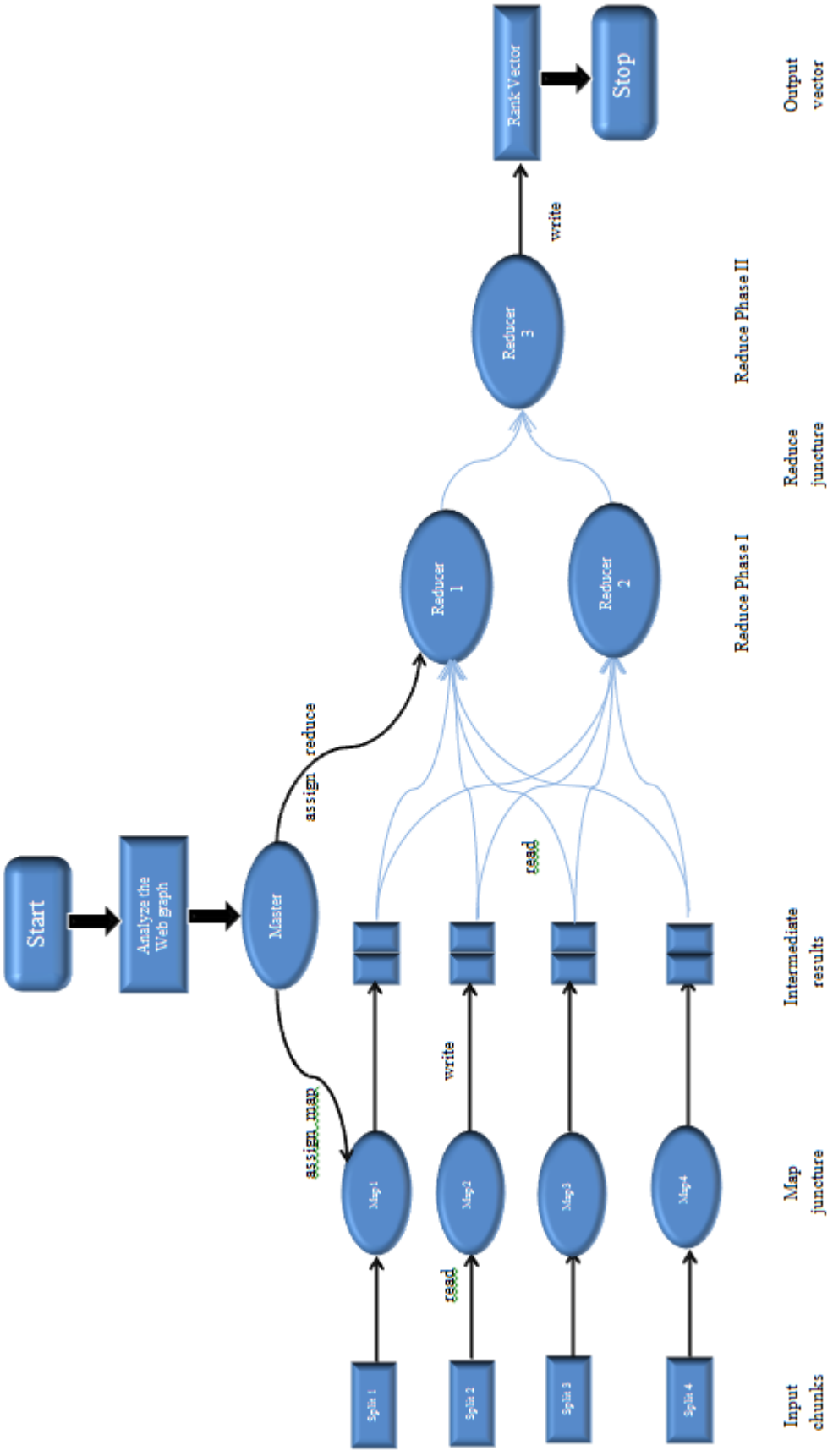

FIGURE 3.3: Spectral Pagerank computation on Map Reduce framework.

3.1.3 Pagerank computation using spectral analysis on Map reduce framework

The Second Proposed Approach comprise of couple of reducer phase. This approach deals with the concept of Spectral analysis for the computation of *eigenvector* of matrix. The approach can be formalized in algorithmic form as follows:

Algorithm 5 Spectral pagerank algorithm on Mapreduce framework

Require: $G = (V, E)$

Ensure: Rank(V)

1. Given a graph G, where V is the vector and E is the edges in adjacency Matrix.
2. Analysis of $G = (V, E)$
3. Partition of G in S_n chunks.
4. $map(S_1, PR_1)$
   return $list(S_1, PR_1)$
5. Reducer Phase I $reduce(S_1, list(PR_1))$
   return $list(SIV_1)$
6. Reducer Phase II $reduce(list(SIV_1))$
   return $Rank(V) \in largesteigenvalue$

Description: Consider a directed graph $G = (V, E)$ such that, V is the vertices and E is the edges in adjacency Matrix of web pages. After the analysis of the matrix and it is divided into n chunks say $S_1, S_2, ...S_n$ of equal sizes. Now these chunks assigned to mapper function. Mapper function will calculate the Pagerank Matrix equation PR. After that various mapppers will send their intermediate results to *reducer*. The *reducer* process is divided into two phase here. The Reducer phase I deals with sorting intermediate results given by mappers and calculate the *Spectral eigen vector*, SIV_1. A list of several *Spectral eigen vector*, $list(SIV_1)$ will be sent to second Reducer phase. Reducer phase II will compare the list and outputs the largest *Spectral eigen vector*, SIV which will be the rank vector $Rank(V)$.

The concept of spectral analysis to compute eigenvalue, as described in introduction section, gives the following relation:

$$\lambda_1 = \max_Z \frac{Z^T M Z}{Z^T Z}$$

, where Z is a random vector having normalized values between 0 to 1 and λ_1 represent the maximum eigenvalue.[2]

We have used a hierarchy of reducer function as shown in Figure 3.2. There are two levels of reducer function are there. At first level we have two reducer function namely *reducer*1 and *reducer*2. Both these functions will receive the intermediate results from various *mapper* functions and compute their corresponding eigenvalue and eigenvector. A set of 500 random vectors have been fed to each *reducer* function, *reducer*1 and *reducer*2. The vector which will return the maximum eigenvalue among 500 random vectors will be the returned to the third *reducer* function *reducer*3. Now intermediate results form both the function will be processed by *reducer*3 which will further outputs the highest eigenvalue and corresponding eigenvector. The eigenvector returned by *reducer*3 will be the rank vector of the input graph.

The results of both the approaches and the time taken are compared with traditional pagerank algorithm. The results of pagerank cumputation on map reduce are computed exactly similar to traditional pagerank algorithm with considerable amount of time saved. The results of spectral analysis based approach when compared with traditional pagerank algorithm lies between 75% to 95% accurate. But the time difference between the two algorithms are exceptionally different, the remarkable amount of time has been saved. The spectral analysis based approach saved $80\% - 90\%$ of the time as compared to pagerank algorithm. To check the accuracy between results we have used the cosine similarity function.According to Euclidean product [15] the cosine of two vectors is:

$$a \cdot b = ||a||||b||cos\theta$$

The cosine similarity between two vectors A and B is given by [16]:

$$similarity = cos\theta = \frac{A \cdot B}{||A||||B||}$$

3.1.4 Processes

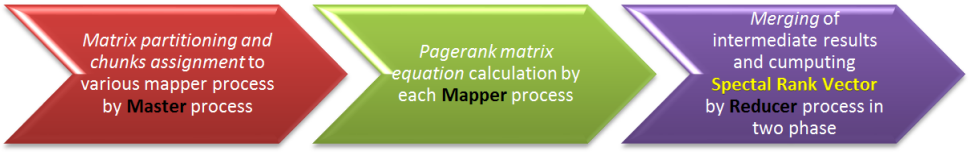

FIGURE 3.4: Process flow in Spectral pagerank algorithm on Mapreduce framework

Three different processes comprises the whole system. The algorithm starts with *Master* process which assigns task to various *Mapper* process and in last *Reducer* process will feed intermediate results from every *mapper* to produce final output. The functioning of these processes in this algorithm are as follows:

1. *Master* process: The transition matrix of the given directed will be the input of *master* process. The transition matrix will be *partitioned* into equal size chunks here and each of these chunks will be *assigned* to an idle *mapper*.

2. *Mapper* process: Assigned chunks will be processed here by *mapper* process to compute the *Pagerank matrix equation* which will be termed as intermediate results. The intermediate results will now be forwarded to *reducer* process.

   The Reducer phase I deals with sorting intermediate results given by mappers and calculate the *Spectral eigen vector*, SIV_1. A list of several *Spectral eigen vector*, $list(SIV_1)$ will be sent to second Reducer phase.

3. *Reducer* process: The hierarchy *Reducer* process is used here and is divided into two phase, Reducer phase I and Reducer phase II. All the intermediate results i.e. *Pagerank matrix equation* from various *mappers* will now be *merged* together. Once the whole intermediate results are merged at reducer

phase I, the user defined function to calculate the *Spectral eigen vector*, SIV_1 of regenerated matrix will be applied. A list of 500 *Spectral eigen vector*, $list(SIV_1)$ each by two *reducers*, at Reducer phase I, will be sent to Reducer phase II. Reducer phase II will compare the list and outputs the largest *vector*. This largest *vector* is the output of the algorithm i.e. **Spectral Rank Vector**.

CHAPTER 4

IMPLEMENTATION DETAILS AND RESULTS

4.1 Platform

- The proposed work has been implemented in R language, R is a programming language and software environment for statistical computing and graphics. It compiles and runs on a broad variety of Windows, UNIX and Mac OS platforms.
- The R language is very popular among statisticians and data miners for developing statistical software and data analysis.
- Ross Ihaka and Robert Gentleman developed this language at the University of Auckland, New Zealand. R is named partly after the first names of the first two R authors.
- R is a very useful and light weight tool for data mining and processing purpose. One of R's potency is the amenities with which well-designed publication-quality plots can be generated, in addition mathematical symbols and formulas where required.
- The source code for the R software environment is written primarily in C, Fortran, and R.
- R is open source language under the GNU General Public License, and precompiled binary versions are provided for various operating systems.
- We have used 64 bit operating system windows 7 having processor with 2.30GHZ frequency working on 4GB RAM.

- R provides a wide variety of statistical and graphical techniques, including time-series analysis, classification, clustering, and others.
- Open source GUI interface, R Stdio is the software to perform computations in R

4.2 Dataset

We have used simulated data first to evaluate our proposed work. Figure 4.1. is the example of one of the various size directed graph used to observe results of our work. Its a directed graph of node size 100. We have also used google web graph dataset ,downloaded from Stanford Network Analysis Project website [17]. We have also created random binary adjacency matrix of varying size . The adjacency matrix represent the directed graph. The entries of matrix are 1 and 0 representing the presence of link between two nodes, 1 indicates the presence of link else 0. The varying size of matrix is used for comparison and analysis on results of proposed approaches in contrast to traditional pagerank algorithm .

4.3 Description

The implementation starts with the analysis of web graph. The web graph is then converted into transition matrix where vertices/nodes represent the web pages. If a page has link to any other page represented by 1 else by 0.

The matrix is then partitioned into equal size chunks as described in Figure 3.1.These chunks of web graph will be the input for map functions. Further chunks are assigned to different mappers, for the computation of intermediate PageRank matrix. Now every mapper function will send their intermediate results to reducer function which will further compute the PageRank vector. To make the algorithm even faster we have used the concept of Spectral analysis for the calculation of largest eigenvector of the given graph.

4.4 Results

We have performed computation and analysis on different size of directed graphs. Variable size of data sets have been used to excavate patterns from our results. We have implemented both of our approaches and also the traditional pagerank algorithm to compare the results from it. All the results are computed in R language. The graphs also generated using R language in R Studio. The results and there comparison with traditional pagerank algorithm are as follows:

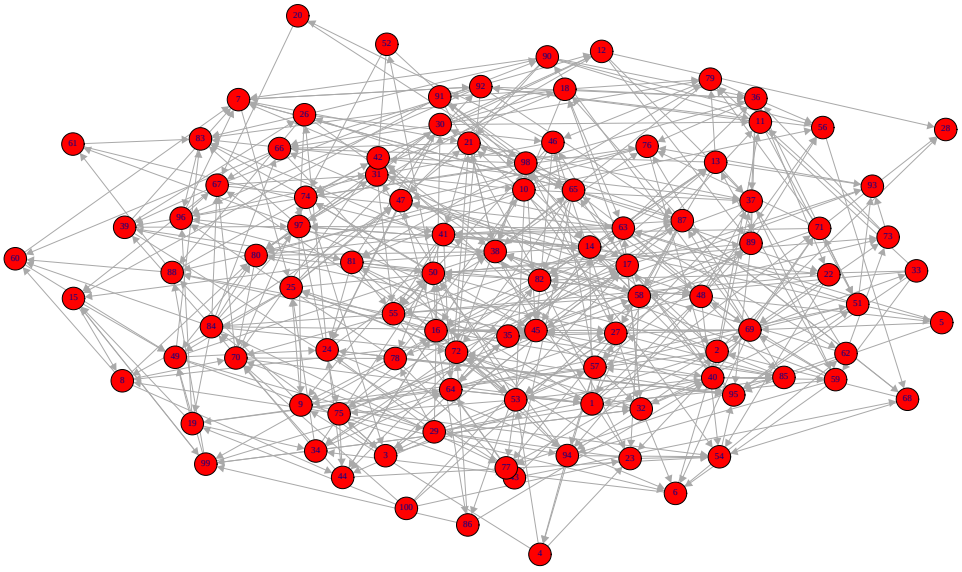

FIGURE 4.1: Directed graph with 100 nodes

4.4.1 Pagerank computation on mapreduce framework

The computation of our algorithm requires a directed graph as a input. The vertices of directed represents page/node and edges between two nodes represents the directed link between them. These links serves are the basis of endorsement that a page give to another. Figure 4.1 is the example of a directed graph that we have created in R studio. The size of directed graph in Figure 4.1. is 100 nodes and 500 edges. The transition for same graph will be of size $100 * 100$. These transition matrix is divided into 4 chunks of size $25 * 25$ and each chunk has been assigned to a *mapper* function. The collective results from each *mapper* function is then processed by *reducer* function. The result generated by *reducer* function corresponding to above graph Figure 4.1. by our first approach, Pagerank computation on mapreduce framework, i.e. the rank vector is shown in Table I. The

Index	Rank
1	0.021865397
2	0.010075798
3	0.008116314
4	0.006808011
5	0.007312947
6	0.015225163
7	0.024353988
8	0.014635842
9	0.006767254
10	0.02090701
11	0.011939445
12	0.005560608
13	0.012121461
14	0.006352905
15	0.007826322
:	:
:	:
:	:
98	0.005089991
99	0.019823114
100	0.0015

TABLE 4.1: MapReduce Rank vector

Index	Rank
1	0.021865397
2	0.010075798
3	0.008116314
4	0.006808011
5	0.007312947
6	0.015225163
7	0.024353988
8	0.014635842
9	0.006767254
10	0.02090701
11	0.011939445
12	0.005560608
13	0.012121461
14	0.006352905
15	0.007826322
:	:
:	:
:	:
98	0.005089991
99	0.019823114
100	0.0015

TABLE 4.2: PageRank vector

Index	Rank
1	0.010395083
2	0.005409302
3	0.004750988
4	0.007859531
5	0.012075618
6	0.003609199
7	0.016540683
8	0.008464667
9	0.001267025
10	0.005864148
11	0.001650276
12	0.000212519
13	0.001240579
14	0.0017786997
15	0.000907024
:	:
:	:
:	:
98	0.013422966
99	0.011004575
100	0.003075954

TABLE 4.3: Spectral Rank vector

results are compared with traditional pagerank algorithm in Table II. The results of Pagerank computation on mapreduce framework approach came similar to that traditional pagerank algorithm. This approaches seems to be saving sufficient time quantum with no information loss.

4.4.2 Spectral pagerank algorithm on Mapreduce framework

The second approach is based on spectral analysis for computation of maximum eigenvalue and corresponding eigenvector. The system design is also changed in this approach where the reducer phase is split into 2 levels as depicted in Figure

Nodes	**Pagerank**	**MR-Pagerank**	**Spectral-Pagerank**
2000	86.90	80.48	36.02
4000	513.97	452.16	122.69
6000	1516.47	1348.78	388.78
8000	3962.53	3178.81	609.72
10000	8048.20	6016.63	843.12
12000	14043.40	9019.32	1173.99
14000	22410.69	13561.28	1622.83

TABLE 4.4: Time of execution in secs

3.3. At level 1 two *reducer* processes are assigned the resultant matrix from various *mappers* and each *reducer* process will compute be fed with 500 random vectors. Now both processes will work in parallel to compute their the maximum eigenvalue. Result of both the *reducer* process will be compared at 2 level of reducer phase and the highest among among them will be the *rank* vector. An example where we computed *rank* vector of the directed graph, Figure 4.1. The results of Spectral based *rank* vector (Table III), are then compared with tradition pagerank algorithm (Table II). The comparison between two vectors is done via cosine similarity, stated above in introduction section. The cosine similarity score between table II & table III is 0.88730 i.e. 88.73% similar. The difference among time of execution between our approaches and traditional pagerank algorithm is very encouraging. The time line for different size of directed graph comparing all 3 algorithms is described in next section.

4.5 Analysis

The directed graph size ranging upto 14000 nodes i.e. the transition matrix of range upto $14000 * 14000$ is used to excavate the pattern of results from both the approaches. The results of first approach, pagerank computation on mapreduce framework, in table I matches exactly with traditional pagerank score in table II and in addition a considerable amount of time is saved as well. The cosine similarity score between results of our second approach, Spectral pagerank algorithm on

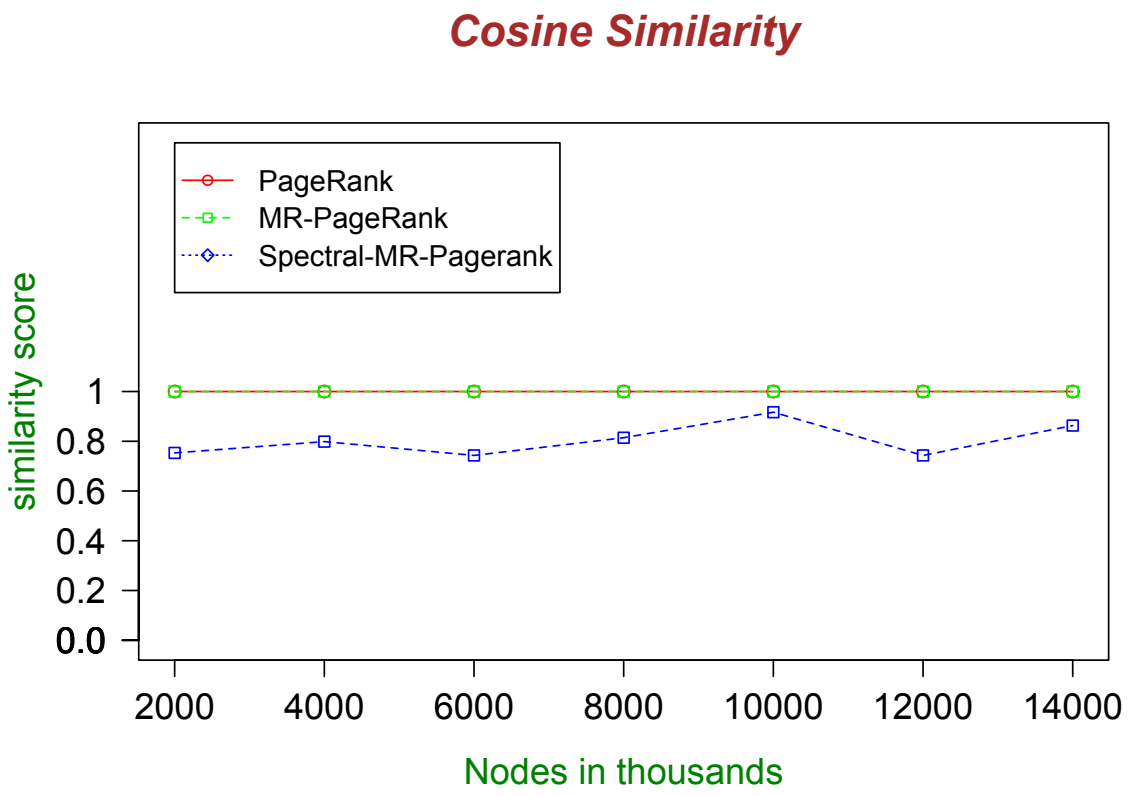

FIGURE 4.2: Cosine similarity between PageRank, Mapreduce PageRank and Spectral Mapreduce pagerank algorithms

mapreduce framework, (in table III) and results traditional pagerank algorithm(in table II) ranges between 0.7431 to 0.9168 i.e. $74\% \rightarrow 91\%$. But when it comes to time of execution the exceptional amount of time is saved by our second approach, Spectral pagerank algorithm on mapreduce framework. The time of execution of all the 3 algorithms for graph size ranging $2000 \rightarrow 14000$ is depicted in table IV. Particular row in table IV represents the time taken by individual algorithm, for a particular node size, in seconds. It can be seen clearly that the difference in time between proposed approaches and the traditional algorithm is promising. To understand the variance in behavior of these three algorithms more preciously, we have plotted the graph as per there results. There are two parameters taken in consideration to evaluate results of our proposed approaches.

- The cosine similarity.
- The time of execution.

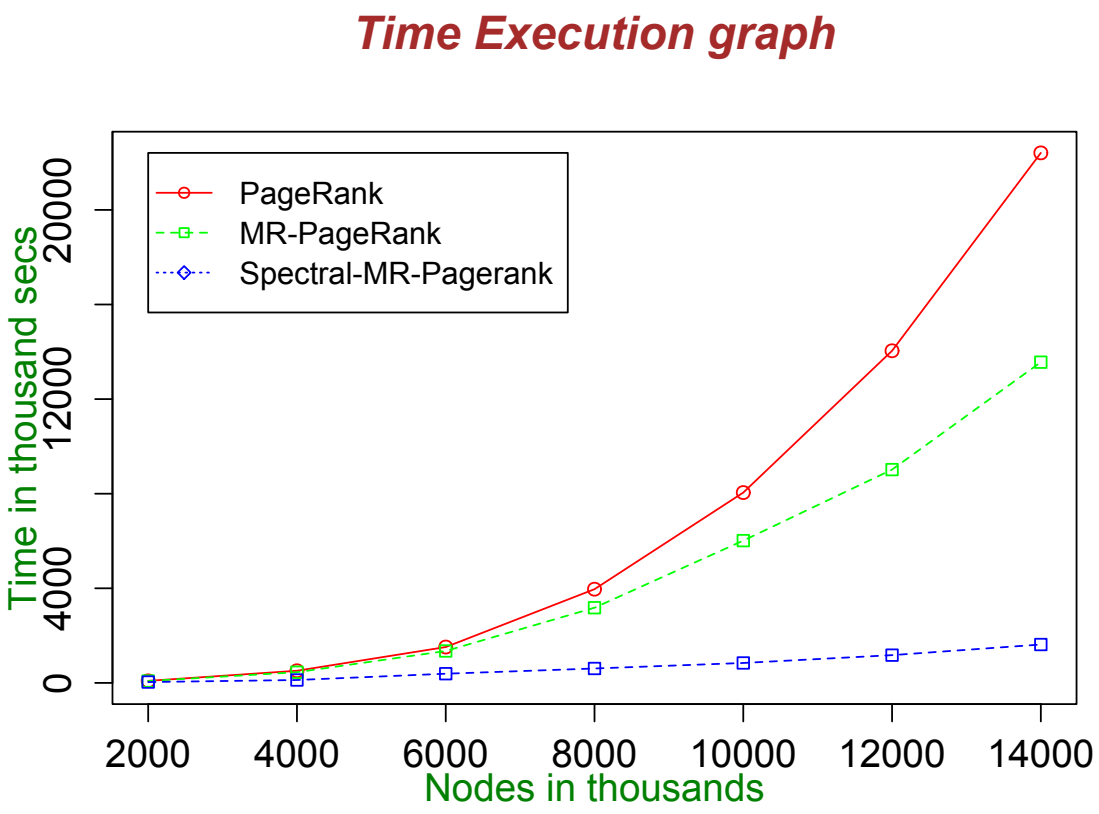

FIGURE 4.3: Time execution graph between PageRank, Mapreduce PageRank and Spectral Mapreduce pagerank algorithms

Figure 4.2. depicts the cosine similarity between three specified algorithms in which x axis represents nodes in thousands and y axis represents the cosine similarity score of the algorithms. The cosine similarity score of computed between proposed approaches and the traditional pagerank algorithm. In Figure 4.2. red line represents the traditional pagerank algorithm, green line represents our first algorithm i.e. Pagerank computation on mapreduce framework and blue line represents our second algorithm i.e. Spectral pagerank algorithm on mapreduce framework. As you can see the green line overlaps over red line, this indicates that the results of our first approach are 100% accurate in comparison to traditional pagerank algorithm. The blue line has deflection between 0.7431 to 0.9168 which says the results of Spectral pagerank algorithm on mapreduce framework has 74% $\rightarrow$ 91% accuracy as compared to traditional pagerank algorithm.

In Figure 4.3. which is the time execution graph the difference of time taken for execution by each three algorithms has been framed. The x axis represents nodes

in thousands and y axis represents the time of execution of the algorithms across various node size. Here again the red, green and blue line represents the traditional pagerank algorithm, the Pagerank computation on mapreduce framework and the Spectral pagerank algorithm on mapreduce framework , respectively. As you can clearly that red line is inclining speedily as the number of nodes increasing while green line seems to be saving considerable amount of time. But in case of blue line which is the Spectral pagerank algorithm on mapreduce framework, the remarkable amount of time is saved. The difference between timeline of traditional pagerank and the spectral pagerank on mapreduce framework is exceptional.

CHAPTER 5

APPLICATION

There exist various domain where we can find the application of ranking algorithms. The domains like business, education, search engine optimization and many more. Product ranking, recommend system, best advertisement selection etc are the example where we can use efficient and fast computing ranking algorithm. One of the major issue that scholars face is, searching best publication from huge citation network. One can take the number of citation as the measure of quality but some group of authors cite each other to get high citation score. In addition there exist several journals and conference which allow bogus publications with imitative results. So searching for most relevant publication is a well known problem that we scholar face. The quality of a publication can be judged by number of citation it gets from other quality publications. The citation from a good publication can be taken as measure of endorsement.

We have taken data sets of, High-energy physics theory citation network [18]. The data set is the citation graph consists of 27,770 papers with 352,807 edges. The citation graph has directed edges among nodes. If a publication say u has an citation to publication v, then there is a directed edge from u to v. If a publication cites, or is cited by, a publication exclusive the dataset, the graph does not contain any information about it.

We have computed *rank* vector for this citation graph network using both our approaches. The results are again compared to traditional pagerank algorithm. The results of first approach (table VII) matches exactly with traditional pagerank results (table V). The rank top 10 of publications are listed in the table VII., computed by "Pagerank computation on Mapreduce framework" approach where ID

Rank	ID	Rank Score
1	27749	0.087557354
2	26661	0.087363177
3	27688	0.072259625
4	27728	0.069806224
5	26671	0.069780612
6	27766	0.067153008
7	27551	0.062072703
8	26638	0.029286035
9	26660	0.029250671
10	27636	0.017826953

TABLE 5.1: Mapreduce Rank vector

Rank	ID	Rank Score
1	27749	0.087557354
2	26661	0.087363177
3	27688	0.072259625
4	27728	0.069806224
5	26671	0.069780612
6	27766	0.067153008
7	27551	0.062072703
8	26638	0.029286035
9	26660	0.029250671
10	27636	0.017826953

TABLE 5.2: PageRank vector

Rank	ID	Rank Score
1	27749	0.000197891
2	26661	0.000197831
3	27688	0.000197801
4	27109	0.000197800
5	26671	0.000197793
6	27636	0.000197792
7	27766	0.000197771
8	1422	0.000197743
9	26660	0.000197343
10	4422	0.000197742

TABLE 5.3: Spectral Rank vector

denotes the publication ID and rank score is the value of rank vector corresponding to publication ID. The results of "Spectral pagerank algorithm on Mapreduce framework" approach are listed iin table VI. The results are seems to be close to that of traditional pagerank (in table V). While grouping the publication based on some criteria can have more accurate result using second approach e.g. grouping based on some threshold rank score.

The time taken by both of our approaches were sufficient less in contrast to traditional pagerank algorithm. Figure 5.1. is a bar plot representing time taken by three algorithms to compute rank vector of above discussed citation network graph. The time taken in minutes by traditional pagerank algorithm is represented by red bar, the green bar represents time of execution for our first approach while blue bar represents time taken by the spectral analysis based approach to compute rank of citation network graph. As you can see in Figure 5.1. the performance of

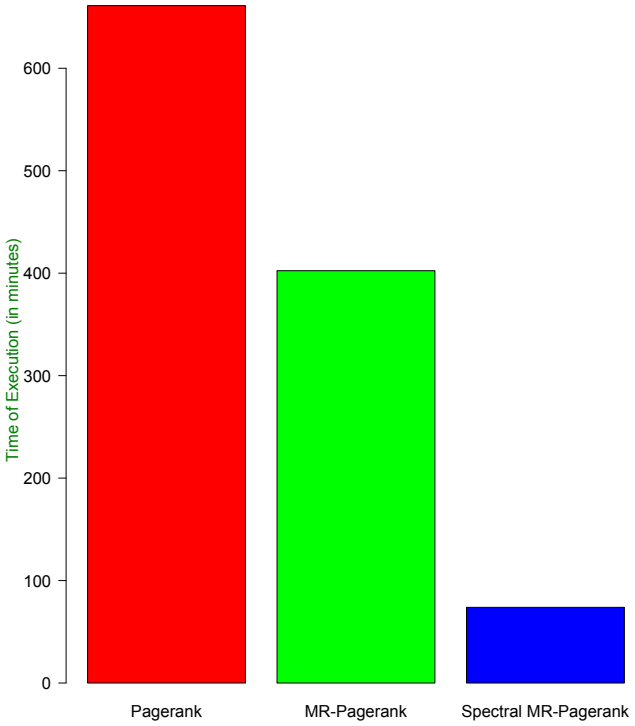

FIGURE 5.1: Time execution bar plot

spectral based approach came out to be best in comparison to other two as far as time is concerned.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

With the enormous processing of huge size directed graph, we conquer many problems like processing of large size graphs, communication among various processes running in parallel, system configuration limitations etc. We have proposed two approaches for distributed, fault tolerant and time efficient ranking system. First is "Pagerank computation on mapreduce framework" and second, "Spectral pagerank algorithm on Mapreduce framework". Both the approaches are time efficient as compared to traditional pagerank algorithm. The first approach i.e. Pagerank computation on mapreduce framework, has 100% accurate results and considerable amount of time saved as compared to traditional pagerank algorithm, hence it can replace traditional algorithm in areas where high accuracy results is the first priority and in addition a distributed, fault tolerant and considerable time saving system, in need .The second approach i.e. "Spectral pagerank algorithm on Mapreduce framework" can be used in areas where time is money and some loss of information can be acceptable e.g. group ranking. On the cost of few information loss this approach guarantees distributed, fault tolerant system with remarkable quantum of time saved.

6.2 Future Work

Till now the Rank vector calculation on Mapreduce Framework and Spectral rank vector calculation on Mapreduce Framework has been done. Further approach is to perform block diagonal approximation of the adjacency matrix. Now we have

number of blocks representing together the whole matrix. Assigning those blocks to *mapper* functions and then following the whole concept of cloud computing i.e Mapreduce computation would be the theme of future work. Another approach would to using logs of the web server for determining the number of hits a web page gets, based on which we evaluate the importance of web page.

BIBLIOGRAPHY

[1] J. P. D. A. Ponce, "Ranking-paginas-web-ecuador," 2015, http://blog.formaciongerencial.com/wp-content/uploads/2015/01/thailand-website-ranking.jpg.

[2] R. Kannan and S. Vempala, "Spectral algorithms," *Found. Trends Theor. Comput. Sci.*, vol. 4, no. 3–4, pp. 157–288, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1561/0400000025

[3] R. Kosala and H. Blockeel, "Web mining research: A survey," *SIGKDD Explor. Newsl.*, vol. 2, no. 1, pp. 1–15, Jun. 2000. [Online]. Available: http://doi.acm.org/10.1145/360402.360406

[4] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," 2004.

[5] M. de Kunder, "The size of the world wide web (the internet)," 2015, http://www.worldwidewebsize.com.

[6] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/324133.324140

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1-7, pp. 107–117, Apr. 1998. [Online]. Available: http://dx.doi.org/10.1016/S0169-7552(98)00110-X

[8] O. Kurland and L. Lee, "Pagerank without hyperlinks: Structural re-ranking using links induced by language models," pp. 306–313, 2005. [Online]. Available: http://doi.acm.org/10.1145/1076034.1076087

[9] M. Benzi, E. Estrada, and C. Klymko, "Ranking hubs and authorities using matrix functions," *CoRR*, vol. abs/1201.3120, 2012.

[10] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," pp. 280–290, 2003. [Online]. Available: http://doi.acm.org/10.1145/775152.775192

[11] X. L. J Zhang, "Full-text and topic based authorrank and enhanced publication ranking," pp. 393–394, 2013. [Online]. Available: http://doi.acm.org/10.1145/2467696.2467748

[12] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," in *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998, pp. 161–172. [Online]. Available: citeseer.nj.nec.com/page98pagerank.html

[13] RalucRemus, "Pagerank algorithm - the mathematics of google search," 2009, http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html.

[14] ——, "Hits algorithm -hubs and authorities on the internet," 2009, http://www.math.cornell.edu/~mec/Winter2009/RalucRemus/Lecture4/lecture4.html.

[15] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, vol. 17, pp. 529–533, 1995.

[16] A. Singhal, "Modern information retrieval: a brief overview," *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, vol. 24, p. 2001, 2001.

[17] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *CoRR*, vol. abs/0810.1355, 2008. [Online]. Available: http://arxiv.org/abs/0810.1355

[18] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005, pp. 177–187. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081893

[19] T. Haveliwala and S. Kamvar, "The second eigenvalue of the google matrix," 2003. [Online]. Available: citeseer.ist.psu.edu/haveliwala03second.html

[20] K. Bryan and T. Leise, "The $ 25,000,000,000 eigenvector: the linear algebra behind google," *SIAM Review*, vol. 48, pp. 569–581, 2006.

[21] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolation methods for accelerating pagerank computations," pp. 261–270, 2003.

[22] A. D. Sarma, S. Gollapudi, and R. Panigrahy, "Estimating pagerank on graph streams," *J. ACM*, vol. 58, no. 3, p. 13, 2011.

[23] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 973–984.

[24] F. Tian and K. Chen, "Towards optimal resource provisioning for running mapreduce programs in public clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 155–162.

[25] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: a runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 810–818.

[26] K. Chakrabarti, D. Xin, and B. Bahmani, "Fast personalized page rank on map reduce," Oct. 7 2014, uS Patent 8,856,047.

[27] F. Can, R. Nuray, and A. B. Sevdik, "Automatic performance evaluation of web search engines," in *INFORMATION PROCESSING AND MANAGEMENT*, 2004, p. 2004.

[28] F. Chung, "A brief survey of pagerank algorithms," 2014.

[29] N. E. Friedkin, "Generalization of the pagerank model," *arXiv preprint arXiv:1401.4740*, 2014.

[30] A. D. Sarma, A. R. Molla, G. Pandurangan, and E. Upfal, "Fast distributed pagerank computation," *Theoretical Computer Science*, vol. 561, pp. 113–121, 2015.

[31] C.-I. Gheorghiu, *Spectral methods for non-standard eigenvalue problems: fluid and structural mechanics and beyond (springerbriefs in mathematics)*. Springer, 2014.

[32] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, "A study of skew in mapreduce applications," *Open Cirrus Summit*, 2011.

[33] J. Lin, "Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail!" *Big Data*, vol. 1, no. 1, pp. 28–37, 2013.