Parallel PageRank Algorithms: A Survey

Atul kumar Srivastava
Department of Computer Science, Institute of Science, Banaras Hindu University, Varanasi, India

Mitali Srivastava
Department of Computer Science, Institute of Science, Banaras Hindu University, Varanasi, India

Rakhi Garg
Computer Science Section, MMV, Banaras Hindu University, Varanasi, India

P. K . Mishra
Department of Computer Science, Institute of Science, Banaras Hindu University, Varanasi, India

Corresponding author: email-mitali.srivastava2011@gmail.com

***Abstract*—**The PageRank method is an important and basic component in effective web search to compute the rank score of each page. The exponential growth of the Internet makes a crucial challenges for search engines to provide up-to-date and relevant user's query search results within time period. The PageRank method computed on huge number of web pages and this is computation intensive task. In this paper, we provide the basic concept of PageRank method and discuss some Parallel PageRank methods. We also compare some Parallel algorithmic concepts like load balance, distributed vs. shared memory and data layout on these algorithms.

Keywords*-PageRank; Power Method; Numerical Method; Parallel PageRank; MPI.*

*****

I. INTRODUCTION

The information and knowledge resources available on the internet continue to increase at exponential rate that required an efficient web search technology for users. For a given user query keyword, web search engine provides millions of results web pages as an answer [1]. Efficient and effective ranking methodology is required to determine the important and relevant web search result. PageRank has been known as most effective and basic method to rank the web pages based on relevancy score [2]. It uses hyperlink structure of web graph to calculate the relevancy score of web page. Since the hyperlink structure data is usually huge, so PageRank computation is still a challenging task for today's web search engines. PageRank of any web page depends upon the PageRank of pages that pointing to this page. A page got higher rank if it is pointed by many high rank web pages [3, 4].

With the tremendous increase of web pages today, computation of PageRank takes a lot of time sometimes it takes several day and consume many computing resources. An efficient and faster PageRank computation method is required because regularly updated web pages directly put impact on frequent re-computation of PageRank method to update the rank value of web pages [5, 6].

Many recent researchers have focuses on some extensive studies to accelerate the PageRank computation. Many accelerating techniques have been proposed in terms of numerical methods, I/O efficient techniques, and in terms of architectural ways [8]. Haveliwal [9] and Chen et al. [8] have used efficient memory computation and disk-based computation. Arasu et al. [10] accelerates the convergence rate of PageRank computation by using Gauss-seidel method and structural properties of web graphs. Kamvar et al. [11, 12] speed up the PageRank computation by periodically off-estimate the non-principle eigenvector from the current iteration. Boldi et al. [13] computed the PageRank vector in main memory by compressing the web graph data. Rungswang et al. [14] have used PC cluster to compute the PageRank method.

In this paper we review Parallel PageRank algorithm based on architecture setup and platform, and data distribution. The configuration of this paper is as follows: Section 2 describe the basic concept of PageRank method. Section 3, elaborate various Parallel PageRank algorithm and how it is implemented. In Section 4, comparative study of various Parallel PageRank algorithm based on different attribute is given. Finally section 5 concludes the paper.

II. PAGERANK METHOD

The basic definition of PageRank is that a page is important if it is pointed by other important pages. Theoretically, PageRank score of a page is the sum of all pages' score that pointing to it. Let a transition probability matrix M of size n is represented as follows [1, 5]:

$$M_{ij} = \begin{cases} 1/O_i \ , & if\ there\ is\ a\ link\ from\ page\ i\ to\ j \\ 0 \quad , & otherwise \end{cases} \quad (1)$$

In equation (1), O_i denote the number of out-degree of page i. If any row in matrix M contains only zero entries *i.e.* out-degree of any page is zero then it is called dangling page. To deal with the dangling page change the transition probability matrix M into new matrix M'. Let u is the uniform teleportation row vector and n is the number of pages in web graph then M' is defined as:

$$M' = M + d.u \quad (2)$$

In equation (2), d is the n dimensional column vector that denote dangling pages defined as:

$$d_i = \begin{cases} 1 & ,if\ O_i = 0 \\ 0 & ,otherwise \end{cases} \quad (3)$$

 ISSN: 2321-8169
470 – 473

$$u = \left[\frac{1}{n}\right]_{1\times n} \quad (4)$$

In equation (2), the second term in right side $d.u$ denote that if the user visit to any dangling page, then they may randomly jump to any other page with probability $1/n$ [2, 3]. The uniqueness of PageRank vector is guaranteed when matrix M' further transformed into M'' on which Power method is applied by following equation:

$$M'' = \alpha M + (1-\alpha)e.u \quad (5)$$

In equation (5), e is the column identity vector. The modified transition matrix M'' is used to compute PageRank vector. Let ρ^k be the n-dimensional column PageRank vector at iteration k, then Power method is used to compute PageRank of web graph by following equation:

$$\rho^{k+1} = M''.\rho^k \quad (6)$$

Equation (6) iterates until it converges to a given tolerance value. Due to huge size of web graph Power method takes several days to compute PageRank score, so there is need of some Parallel platform like MPI, MapReduce or Spark that compute PageRank vector efficiently and effectively. In further section we discuss some Parallel PageRank algorithm that efficiently compute on those platform and performs better result than basic PageRank algorithm.

III. PARALLEL PAGERANK ALGORITHMS

Efficient computation of PageRank is required due to huge size of web graph. Recent researchers discuss some algorithmic and architectural approaches to speed up the computation of PageRank. Here we review some of those Parallel PageRank algorithm and focuses on their pros and cons, platforms, software etc... To compute PageRank vector on Parallel platform first we need to store hyperlink matrix into binary link structure file B that is partitioned into β chunks. Here β is the number of systems in a cluster setup [14, 15]. Let there are T records in binary structure file and file is described as follows:

dest_id	src_id
1	1029
2	101 23
3	276 34 111
4	21 476 234 165
5	234
6	563 876

file B

out_deg
1
5
3
4
1
7

file O

in_deg
1
2
3
4
1
2

file I

Figure 1: Show the structure of file for Parallel PageRank computation

In Figure 1, file B have two attribute i.e. dest_id and src_id that contains incoming and outgoing pages' id e.g. page 1 is pointed by page 1029 and page 2 is pointed by page 101 and page 23. File O, and file I contains number of outgoing link and incoming link of page of file B.

A. Parallel PageRank algorithm on PC Cluster

Rungsawan et al. [14] have proposed Parallel PageRank based on a cluster of β processor that is connected via fast Ethernet network. To speed up the PageRank computation partition the binary link structure file B is partitioned into β parts such that $B_0, B_1, B_2, \ldots, B_{\beta-1}$. The proposed PageRank algorithm is given below:

Algorithm 1: Parallel PageRank Algorithm [14]

1. Initialize each processor with MPI process id $P_0, P_1, P_2, \ldots$
2. $start = \left(\frac{T\times P_i}{\beta}\right) + 1$, and $end = \left(\frac{T}{\beta}\right) \times (P_i + 1)$
3. $\forall j = 1\ to\ T$, $\rho_{src_{id},P_i}(j) = 1/n$, $\alpha = 0.85$ // assign rank of every web page to *1/n*
4. *Repeat Untill convergence*
5. Repeat till end of file B_{P_i}
6. $j = B_{P_i}.dest\_id$ // to obtain index
7. $\rho_{dest\_id,P_j}(j - end) = \frac{(1-\alpha)}{n} + \alpha\left[\sum_{B_{P_i}.in\_deg} \frac{\rho_{src_{id},P_i}\left[B_{P_i}.src_{id}\right]}{outdegree}\right]$
8. end
9. end

Algorithm (1) iterates until convergence is reached and all processors will be synchronized their blocks of PageRank value in $2\log_2\beta$ communication steps [14]. This experimental result show that speed up curve get close to the ideal speed up for the large data size.

B. Parallel Adaptive PageRank algorithm

Armon et al. [15] proposed a parallel PageRank computation on PC Cluster. Kamvar et al. [16] found that PageRank vector for about *2/3* of pages it converges very early than for other pages. So based on this concept they proposed adaptive PageRank method that avoid the overhead of re-computation of already converged web pages and compute converge web page by using following equation:

$$\forall i\ \rho'(i) = \begin{cases} \rho(i) & ,if\ \rho(i)\ converge \\ \alpha\sum_{j\in S_j}\frac{\rho(j)}{O_j} + \frac{(1-\alpha)}{n} & ,otherwise \end{cases} \quad (7)$$

They proposed Parallel PageRank algorithm and combined it with adaptive PageRank computation. The parallel adaptive PageRank algorithm is given below:

Algorithm 2: Parallel Adaptive PageRank algorithms [15]

1. $\forall i\ \rho(i) = 1/n$ // initialize rank of all pages
2. initialize (S_c, S_N) // initialize convergent rank of pages and
3. while end of non-convergent file
4. while B_i is not end of file
5. if $B_i.dest\_id \in S_c$ // check whether page is in set of converged rank
6. $\rho'[L.dest\_id] = \rho[L.dest\_id]()$
7. else
8. for every j=1,2,… in_deg
9. Score+=$\frac{\rho[L.src\_id_j]}{O[L.src\_id_j]}$
10. $\rho'[L.dest_{id}] = \alpha.score + \frac{(1-\alpha)}{n}$
11. }
12. }

From experiment analysis they have showed that adapting PageRank converges faster by elimination of redundant computation needed for already converged pages, and this algorithm is also scalable for large datasets.

C. Parallel PageRank computation by using Gauss-Seidel method

Kohlschütter et al. [17] accelerate the PageRank computation by applying Gauss-Seidel method rather than Power method in Parallel PageRank. They present a method which combines the Gauss-Seidel and Jacobi method that improved the convergence of PageRank algorithm. They had use combined method and simplified by using following equation and implement on parallel platform (8):

$$\rho^*(i) = (1-\alpha)\rho(i) + \alpha \sum_{(j,i)\in L} \frac{\rho^*(j)}{|O_j|} \qquad (8)$$

In equation (8), $\rho^*(i) = \rho^{(m-1,m)}(i)$ where m denotes iteration number, $(i, j) \in webpages$, and $\rho^{(m-1,m)}$ is an intermediate PageRank vector that combines the previous and current iteration vector.

Through experiment they prove that their approach and original method both produce the same rank score, and combined method is more scalable than other parallel PageRank methods due to parallelization of Gauss-Seidel method [20].

D. PPR – MT algorithm

Bundit et al. [18] proposed an efficient Parallel PageRank method based on Multi-threaded model of MPI. They have executed Parallel adaptive PageRank algorithm on two SMP cluster system. In this algorithm MPI is used as a communication model between processes on different systems, and inter-thread method is used for communication between threads. They have partitioned the binary link structure file to distribute it equally among cluster nodes. The file *B* and *I* are partitioned and distributed among the cluster nodes, while the file *O* is not partitioned. They have proposed following PPR-MT algorithms [18]:

Algorithm 3: PPR-MT Algorithm [18]

1. $\rho_i = 1/n$ // assign rank score to every web page
2. Create threads
3. while till converge
4. for every page $p \in B_i$ assign to thread
5. score=0
6. if $p.dest\_id$ converges then
7. $\rho'[p.dest\_id] = \rho[p.dest\_id]$
8. else
9. compute all score+=$\frac{\rho[p.src\_id]}{O[p.src\_id]}$
10. $\rho'[p.src_{id}] = \alpha.score + \frac{(1-\alpha)}{n}$
11. endfor
12. endwhile

On execution of algorithm (3), it was observed that the speed up of given algorithm is about 23 times faster on four machine than to basic PageRank Power method on single machine.

E. Parallel PageRank of CUDA

Tarun et al. [19] presented parallel PageRank algorithm on Cell-BE Processor and CUDA showed that their algorithm runs almost 3 times faster than Xeon dual core 3.0 GHz. To get better performance they have used SPE (Synergistic Processing Element) for all calculation. They had design data structure to store binary link structure file as follows:

- **Node Array [n1|in_deg1|s1|s2|s3|…..|n2|in_deg|s1|s2|s3|….]** that contains first node followed by its in-degree, followed by source nodes, then second node followed by its in-degree and so on.
- **Degree Array [n1|in_deg1|d1|d2|d3|…..|n2|in_deg|s1|s2|s3|….]** that contains first node followed its in-degree, followed by destination nodes.
- Two array *V1* and *V2* that keep rank of nodes at i^{th} *and* $(i+1)^{th}$ iteration.

Based on given data structure they presented following algorithm [4]:

Algorithm 4: Parallel PageRank on CUDA [19]

1. Initialize PageRank to every page $\rho(i) = 1$
2. for every iteration
3. Degree Array and *V1* are equally divided among number of SPE, and rank is calculated on PageRank array.
4. After calculation *V2* is updated with *V1*.

IV. COMPARISON OF PARALLEL PAGERANK ALGORITHM

In this section, we categorizes different parallel PageRank algorithm developed so far on the basis of load balancing technique used, type of parallel and data layout used. The details are given in Table 1:

V. CONCLUSION

From above review, it is clear that there are lots of Parallel algorithms to compute PageRank vector have been proposed so far. As we can observe that algorithm 1, algorithm 2, and algorithm 5 have not used any load balancing technique, they have distributed data horizontally to every node, and algorithm 3 and 4 have distributed data to every node based on size. Earlier MPI was used as popular platform for parallel computation of PageRank method that required too much attention for users to perform every task like load balancing, and data layout. To achieve all aspect i.e. load balancing, type of parallelism and data layout efficiently is a complex task in MPI. So today there are other platform available like Spark that have greater advantage over MPI. There is a big scope of PageRank computation on other platform like Spark to achieve better efficiency and scale-up.

Table 1: Comparison of Various Parallel PageRank algorithm based different parameter

	Load Balancing	Memory Used		Type of Parallelism		Data Layout	
		Distributed	Shared	Data	Task	Horizontal	Vertical
Parallel PR on PC Cluster [14]	No	Yes	No	Yes	No	Yes	No
Parallel Adaptive PR [15]	No	Yes	No	Yes	No	Yes	No
Parallel PR using GSeidel [17]	Yes	Yes	No	Yes	No	Yes	No
PPR – MT algorithm [18]	Yes	Yes	Yes	Yes	No	Yes	No
Parallel PR on CUDA [19]	No	Yes	No	Yes	Yes (Thread level)	Yes	No

REFERENCES

[1] S. Brin, L. Page (1998), "The Anatomy of a Large-scale Hyper textual Web Search Engine" Proceedings of the Seventh International World Wide Web Conference, Page(s):107-117.

[2] Langville, A. N., and Meyer, C. D. (2004) "Deeper inside pagerank" Internet Mathematics, Vol.1 No.3, pp. 335-380.

[3] Pavel Berkhin (2005), "A survey on PageRank computing", Internet Mathematics 2, Vol.1, Page(s):73–120.

[4] P Desikan, J Srivastava, Vipin Kumar, and Pang-Ning Tan(2002), "Hyperlink Analysis: Techniques and Applications" Page(s):1-42.

[5] Pretto, L.: A theoretical analysis of googles PageRank. In: Laender,A.H.F., Oliveira, A.L. (eds.) SPIRE 2002. LNCS, vol. 2476, pp. 131144. Springer, Heidelberg (2002).

[6] Langville, A.N., Meyer, C.D.: Googles PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton (2006).

[7] Langville, A. N., & Meyer, C. D. (2004, May). Updating pagerank with iterative aggregation. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters (pp. 392-393). ACM.

[8] Chen, Y. Y., Gan, Q., & Suel, T. (2002, November). I/O-efficient techniques for computing PageRank. In Proceedings of the eleventh international conference on Information and knowledge management (pp. 549-557). ACM.

[9] Haveliwala, T. (1999). Efficient computation of PageRank. Stanford.

[10] Arasu, Arvind, et al. "PageRank computation and the structure of the web: Experiments and algorithms." Proceedings of the Eleventh International World Wide Web Conference, Poster Track. 2002.

[11] Kamvar, Sepandar, et al. "Exploiting the block structure of the web for computing pagerank." Stanford University Technical Report (2003).

[12] Kamvar, S. D., Haveliwala, T. H., Manning, C. D., & Golub, G. H. (2003, May). Extrapolation methods for accelerating PageRank computations. In Proceedings of the 12th international conference on World Wide Web (pp. 261-270). ACM.

[13] Boldi, Paolo, and Sebastiano Vigna. "The webgraph framework I: compression techniques." Proceedings of the 13th international conference on World Wide Web. ACM, 2004.

[14] Rungsawang, A., & Manaskasemsak, B. (2003, September). PageRank computation using PC cluster. In European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting (pp. 152-159). Springer Berlin Heidelberg.

[15] Rungsawang, A., & Manaskasemsak, B. (2006, February). Parallel adaptive technique for computing PageRank. In Parallel, Distributed, and Network-Based Processing, 2006. PDP 2006. 14th Euromicro International Conference on (pp. 6-pp). IEEE.

[16] Kamvar, S., Haveliwala, T., & Golub, G. (2004). Adaptive methods for the computation of PageRank. Linear Algebra and its Applications, 386, 51-65.

[17] Kohlschütter, C., Chirita, P. A., & Nejdl, W. (2006, April). Efficient parallel computation of pagerank. In European Conference on Information Retrieval (pp. 241-252). Springer Berlin Heidelberg.

[18] Manaskasemsak, Bundit, Putchong Uthayopas, and Arnon Rungsawang. "A mixed MPI-thread approach for parallel page ranking computation." On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE (2006): 1223-1233.

[19] Kumar, T., Sondhi, P., & Mittal, A. (2012, February). Parallelization of PageRank on multicore processors. In International Conference on Distributed Computing and Internet Technology (pp. 129-140). Springer Berlin Heidelberg.