International Journal of Chemistry and Applications.
ISSN 0974-3111 Volume 3, Number 2 (2011), pp. 201-208
© International Research Publication House
http://www.irphouse.com

A Survey on Distributed Page Ranking

Manju Patel[1] and Shweta Modi[2]

[1]M.E., S.R.I.T, Jabalpur, M.P., India
E-mail: manjupatel05@yahoo.com
[2]HOD, CS/IT, S.R.I.T, Jabalpur, M.P., India
E-mail: modi_sweta84@yahoo.com

Abstract

Page ranking is the fundamental requirement of search engines to make the search results up-to-date the search need to be very fast but the recent growth in world wide web making it difficult for standalone systems because for handling enormous amount of web data system needs tremendous amount of memory & very high speed processors hence other type of systems were proposed which a system calculates the page rank of only of the small part of data and several other processor executes on different parts of data then they exchange the information from each other using some kind of communication system this system reduces the burden on single system by reducing the amount of data on each system. This method is called the distributed system because it computes at different systems & then merges the results as single unit. Although the distributed system designing for page rank calculation is not as simple because the page rank is calculated by using iterative loop hence many approaches are proposed by different researchers in this paper we are discussing some of those.

Keywords: Distributed Processing, Page Ranking, P2P Network, Search Engine.

Introduction

Centralized Internet search engine systems face many challenges like a successful search engine system requires a large data cache with tens of thousands of processors to create inverted text indices, to measure page quality, and to execute user queries. Also, centralized systems are vulnerable to point failures and network problems, and thus must be replicated. For example, Google employs a cluster of more than 15,000

PCs and replicates each of its internal services across multiple machines and multiple geographically distributed sites [1]. Distributed page ranking is one of the proposed solutions for page ranking of large data sets like World Wide Web, but as we know that most of the page ranking algorithms are iterative in nature hence it is important to distribute the pages to each processing unit such that there must be minimum data exchange needed after each iteration loop here we are proposing the simple & very fast technique to optimally cluster the pages in required number of groups.

This paper discusses page ranking technique for a distributed Internet search engine framework. With such a framework, there are no dedicated centralized servers. Instead, every web server participates as an individual search engine over its own (local) data so that crawlers are no longer needed. User queries are processed at related web servers and results will be merged at the client side. There are several partitioning methods are available for distributed page rank calculation but one of the most widely used formulation is proposed by Kamvar [1]. In this formulation, which is based on the power method, the kernel operations are sparse-matrix vector multiply and linear vector operations. Recently Bradley [2] utilized the hypergraph partitioning based models proposed by Catalyurek and Aykanat [3][4] directly for parallel PageRank computation. Unfortunately, because of the huge size of the Web graph, hypergraph partitioning based models are not scalable, when applied directly over the Web matrix. Even though the computations reported in2) are fairly fast; the preprocessing time for partitioning takes even longer than the sequential PageRank computation. To avoid this problem, Cevahir et al.6) suggested sitebased hypergraph partitioning models which reduces the sizes of the hypergraphs used in partitioning, considerably. In addition to reduced preprocessing time, they offer parallelization of the overall iterative algorithm including the linear vector operations and norm operations as well as matrix-vector multiplications for load balancing in the partitioning model, whereas Bradley et al. only consider matrix-vector multiplies. The proposed site-based partitioning scheme reduces the preprocessing time drastically compared to the page-based scheme while producing better partitions in terms of communication volume.

Distributed PageRank for P2P Systems

ShuMing Shi, Jin Yu, GuangWen Yang, DingXing Wang [4] proposed method to perform page ranking in a peer-to-peer environment. Assume there are *K* nodes (called *page rankers*) participating in page ranking, and each of them is in charge of a subset of the whole web pages to be ranked. Pages crawled by crawler(s) are partitioned into *K* groups and mapped onto *K* page rankers according to some strategy. Each page ranker runs a page ranking algorithm on it. Since there have links between pages of different page groups, page rankers need to communicate periodically to exchange updated ranking values. Some key problems will be discussed in this section.

Web Page Partitioning

Different strategies can be adopted to divide web pages among page rankers: divide pages randomly, divide by the hash code of page URLs, or divide by the hash code of websites. As crawler(s) may revisit pages in order to detect changes and refresh the downloaded collection, one page may participate in dividing more than one time. The random dividing strategy doesn't fulfill this need for taking the risk of sending a page to different page rankers on different times. When performing page ranking, page scores may transmit between page rankers, causing communication overhead between nodes. Because number of inner-site links overcomes that of inter-site ones for a web site ([16] finds that 90% of the links in a page point to pages in the same site on average), divide at site granularity instead of page-granularity can reduce communication overhead greatly. To sum up, dividing pages by hash code of websites is a something better strategy.

Distributed PageRank Algorithms

Two different algorithms, DPR1 and DPR2, are shown (see Algorithm 3 and 4) to performing distributed page ranking. Both of them contain a main loop, and in each loop, the algorithm first refreshes the value of *X* (for other groups may have sent new ranks by the afferent links of the group), and then compute vector *R* by one or more iteration steps, and lastly, compute new *Y* and send it to other nodes.

Note that each node runs the algorithm asynchronously, in other words, ranking programs in all the nodes can start at different time, execute at different 'speed', sleep for some time, suspend itself as its wish, or even shutdown. In fact, we can insert some delays before or after any instructions.

```
function DPR1() {
    $R_0 = S$
    $X = 0$
    loop
            $X_{i+1}$=Refresh $X$
            $R_{i+1}$ = GroupPageRank($R_i$,$X_{i+1}$)
            Compute $Y_{i+1}$ and send it to other nodes
            Wait for some time
    while true
}
```

Algorithm 1: Distributed PageRank Algorithm: DPR1.

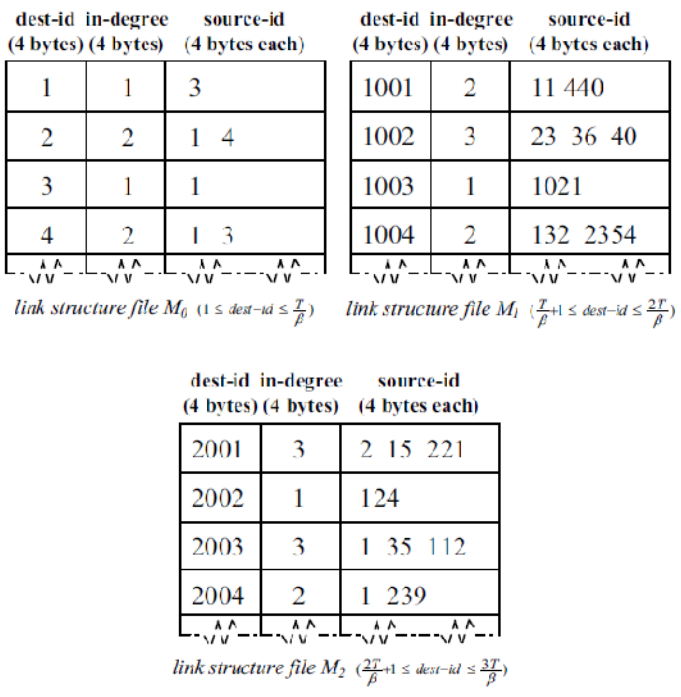
The difference between algorithm DPR1 and DPR2 lies in the style and frequency of refreshing input vector X and updating output vector Y. In each loop of algorithm DPR1,

```
function DPR2() {
    $R_0 = S$
    $X = 0$
    loop
        $X_{i+1}$=Refresh X
        $R_{i+1} = AR_i + \beta E + X_{i+1}$
        Compute $Y_{i+1}$ and send it to other nodes
        Wait for some time
    while true
}
```

Algorithm 2: Distributed PageRank Algorithm: DPR2.

Parallel PageRank Computation on a PC Cluster

Proposed by Bundit Manaskasemsak, Arnon Rungsawang [5] To accelerate the PageRank computation using a cluster of β processors (i.e., β machines), we first partition the binary link structure file *M* into β chunks: M_0, M_1, …., M_{β} *-1*, such that each M_i contains only records referring to destination URLs from the (i*T / β + 1) to the ((i+1)*T / β) where *T* is the total number of URLs in the web graph. Figure 1 as follows illustrates textually the example of the partitioned binary link structure file *Mi*.



dest-id (4 bytes)	in-degree (4 bytes)	source-id (4 bytes each)
1	1	3
2	2	1 4
3	1	1
4	2	1 3

link structure file M_0 $(1 \le dest\text{-}id \le \frac{T}{\beta})$

dest-id (4 bytes)	in-degree (4 bytes)	source-id (4 bytes each)
1001	2	11 440
1002	3	23 36 40
1003	1	1021
1004	2	132 2354

link structure file M_1 $(\frac{T}{\beta}+1 \le dest\text{-}id \le \frac{2T}{\beta})$

dest-id (4 bytes)	in-degree (4 bytes)	source-id (4 bytes each)
2001	3	2 15 221
2002	1	124
2003	3	1 35 112
2004	2	1 239

link structure file M_2 $(\frac{2T}{\beta}+1 \le dest\text{-}id \le \frac{3T}{\beta})$

Figure 1: The partitioned binary link structure file used.

In parallel PageRank Algorithm. During the iterative computing of PageRank scores for a large web graph in parallel, every processor which participates in the pool of tasks is assigned with a process identifier *Pi* = 0, 1, 2, …, β-1. Each processor *Pi* has to allocate, for its own, two separated arrays of floating points: *ranksrc,Pi*, having *T* entries, records all source rank scores of the iteration *jth*, and *rankdest,Pi*, having *T/β* entries, records the local destination rank scores of the iteration (*j*+1)th; and an array of integer *outarry Pi*, having *T/β* entries, records the out-degree of each destination URL. The proposed parallel PageRank computation can be illustrated in Algorithm 3. According to the Algorithm 3, we first assign an MPI process identifier to each machine, calculate the corresponding starting URL (*B*) and the ending URL (*E*), initialize each source rank score (i.e., *ranksrc,Pi*) with *T* 1, and set the damping factor 0 to 0.85. The iterative processes (see line 4-11 in Algorithm 1) repeat until all final PageRank scores convert. By experiments, those scores convert with L1 norm of residual errors [6] < 0.025 when the iterative processes repeat at least 50 times. In the following experimental results, we then report with the number of iteration pass set to 50.

```
1 : assign MPI process identifier to each machine, $P_i = 0, 1, 2, \dots$
2 : $B = \left(\frac{T}{\beta} \times P_i\right) + 1;\ \ E = \frac{T}{\beta} \times (P_i + 1);$
3 : $\forall_{t=1..T} Rank_{src,Pi}[t] = \frac{1}{T};\ \ \alpha = 0.85;$
4 : for $round = 1$ to 50 do
5 :     while ($M_{Pi}$ is not end of file) do
6 :         $t = M_{Pi}.dest\text{-}id;$
7 :         $Rank_{dest,Pi}[t - B] =$
               $(1-\alpha) + \alpha \left( \sum_{M_{Pi}.in\text{-}degree} \frac{Rank_{src,Pi}[M_{Pi}.source\text{-}id]}{N_{Rank_{src,Pi}[M_{Pi}.source\text{-}id]}} \right);$
8 :     end while
9 :     each process has to synchronize all local $Rank_{dest,Pi}$
10 :    $\forall_{t=B..E} Rank_{src,Pi}[t] = Rank_{dest,Pi}[t - B];$
11 : end for
```

Algorithm 3: Distributed PageRank Algorithm: DPR3.

Computing PageRank in a Distributed Internet Search System

Proposed by Yuan Wang David J. DeWitt [6] The topology of the web linkage structure suggests that connectivity-based page importance measures can be computed at individual web servers, i.e., every web server can independently compute a "*Local PageRank*" vector over its local pages. Since the majority of links in the web link graph are intra-server links, the relative rankings between most pages within a server are determined by the intra-server links. So the result of local query execution is likely comparable to its corresponding sub list of the result obtained using the global *PageRank* algorithm. The inter-server links can be used to compute "*ServerRank*", which measures the relative importance of the different web servers. Both *Local*

PageRank and *ServerRank* are used in combination to merge query results from multiple sites into a single, ranked hyperlink list.

The outline of the algorithm follows:

1. *Each web server constructs a web link graph based on its own pages to compute its "Local PageRank" vector.*
2. *Web servers exchange their inter-server hyperlink information with each other and compute a "ServerRank" vector.*
3. *Web servers use the "ServerRank" vector to refine their "Local PageRank" vectors, which are actually used for local query execution.*
4. *After receiving the results of a query from multiple sites, the submitting server uses the "Server- Rank" vector and the "Local PageRank" values that are associated with the results to generate the final result list. Each step is described in detail in the following sections. Notice, that for static data sets, both the Local PageRank vectors and the ServerRank vector need to be only computed once. As shown later, all algorithms are efficient and can be exercised frequently in case of updates.*

Conclusion

In the above study we discussed three methods of distributed page ranking for three different networks (P2P, Giga bit pc cluster & over internet environment) everyone has their own benefits & drawbacks some of those are compared below.

Figure 2 shows linear speed up proportionate to number of processors in Giga bit PC cluster system other system does not performs like this because of large communication delay between different unit & larger overheads.

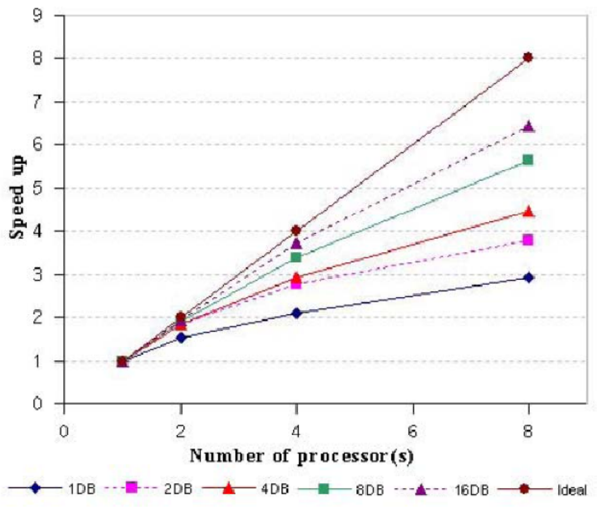


Figure 2: Speedup curves concluded from the experiments in Giga bit PC cluster system.

Although the reliability of other two system are much better than the Giga bit cluster because of the considerations taken about broken connections & asynchronous operations

Future Work

Distributed page ranking are needed because size of the web grows at a remarkable speed and centralized page ranking is not scalable. PageRank can be modified slightly for open systems. To do page ranking distributedly, pages can be partitioned by hash code of their websites. Distributed PageRank converges to the ranks of centralized PageRank. Indirect transmission can be adopted to achieve scalable communication. The convergence time is judged by network bisection bandwidth and the bottleneck bandwidth of nodes. Future works include: Doing more experiments (and using larger datasets) to discover more interesting phenomena in distributed page ranking. And explore more methods for reducing communication overhead and convergence time.

References

[1] S. Kamvar, T. Haveliwala, C. Manning and G. Golub, Extrapolation Methods for Accelerating PageRank Computations, *Proc. of the 12th WWW Conf.*, 2003.

[2] S. Kamvar, T. Haveliwala and G. Golub, Adaptive Methods for the Computation of PageRank, *Linear Algebra and its Applications, Spec. Issue on the Numerical Sol. of Markov Chains*, November 2003.

[3] http://www.google.com

[4] Distributed Page Ranking in Structured P2P Networks ShuMing Shi, Jin Yu, GuangWen Yang, DingXing Wang

[5] Parallel PageRank Computation on a Gigabit PC Cluster Bundit Manaskasemsak, Arnon Rungsawang Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04) 2004 IEEE

[6] Computing PageRank in a Distributed Internet Search System Yuan Wang David J. DeWittProceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[7] Owe Axelsson. Iterative Solution Methods. Cambridge University Press. 1994

[8] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, etc. Extrapolation Methods for Accelerating PageRank Computations. Stanford University Technical Report, 2002.

[9] T. H. Haveliwala. Topic-sensitive PageRank. In Proceedings of the Eleventh International World Wide Web Conference, 2002.

[10] D. Rafiei and A.O. Mendelzon. What is this page known for? Computing web page reputations. In Proceedings of the Ninth International World Wide Web Conference, 2000.

[11] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In Proceedings of the Eighth International World Wide Web Conference, 1999.

[12] Vipin Kumar, Ananth Grama, etc. Introduction to Parallel Computing, Design and Analysis of Algorithms. The Benjamin/Cummings Publishing Company.
[13] Ratnasamy, S., et al. A Scalable Content-Addressable Network. in ACM SIGCOMM. 2001. San Diego, CA, USA.
[14] Stoica, I., et al. Chord: A scalable peer-to-peer lookup service for Internet applications. in ACM SIGCOMM. 2001. San Diego, CA, USA.