Indian Journal of Science and Technology, *Vol 9(38), DOI: 10.17485/ijst/2016/v9i38/90410, October 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improved Parallel PageRank Algorithm for Spam Filtering

Hema Dubey[1*], Nilay Khare[1], K. K. Appu Kuttan[1] and Shreyas Bhatia[2]

[1]Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal - 462003, Madhya Pradesh, India; hema32150@gmail.com, nilay.khare@rediffmail.com, pp_kuttan@Yahoo.com,
[2]Adobe Systems, Noida – 201304, India; shreyasbhatia09@gmail.com

Abstract

Background/Objectives: PageRanking algorithm is a well known link based technique given by Google for indexing of its web pages. This algorithm works on the linking structure of web pages id est inbound and outbound links of pages. The existing Page Rank algorithm follows equal distribution law that is; it distributes the Page Rank of a web page evenly among all the outgoing links. The problem with the uniform distribution of Page Rank is that sometimes uninteresting pages got high Page Rank values. **Methods/Statistical Analysis**: This paper proposed an improved parallel Page Rank algorithm that un-uniformly distributes the Page Rank values among all the outgoing links. The proposed work has been implemented on NVIDIA Quadro 2000 GPU architecture using CUDA programming language. **Findings:** The proposed algorithm mitigates spam and provides better results in terms of computational time as compared to Parallel Page Rank, because it assigns higher priority to important pages and less priority to less important web pages. By assigning values in such a fashion important pages show an increase in the Page Rank value and unrelated pages that is spam pages show a decrease in Page Rank value. **Application:** The proposed work performs spam filtering by classifying important as well as irrelevant web pages.

Keywords: CUDA, GPU, Non-Uniform Distribution, Parallel Page Rank, Spam Pages

1. Introduction

With the rapidly growing www, search engines play an imperative role in the extraction of relevant information from the internet. The search engine is an important part of any Information Retrieval (IR) system. An index is created by a search engine on the basis of which it matches the query. The index includes the words present in each of the documents, plus the pointers to the locations of words present within the documents, this is known as Inverted File. There are four essential modules present in the search engine[1]:

- A Document Processor.
- A Query Processor.
- A Searching and Matching Function.
- A Ranker.

Figure 1 depicts that Document Processor performs pre-processing, recognize probable indexable items in a document, removing stop words, stemming words, extracting index entries, assigning weights to terms and then finally creating the index. The Query Processor carries out several steps. Some of them are tokenizing the query expressions into comprehensible segments, parsing, obliterate stop words, stemming words, creating query representation, expanding query expression and weighting query expression[1].

When a user fires a query in a search engine, for instance, Google, Yahoo, Baidu, Bing, the search engine returns a million of web pages. Some pages are relevant, important and useful for the user, but some are of less important. So it becomes foremost necessary to rank web pages in accordance with their relevance and pertinence so as to display significant web pages on the top of the search

*Author for correspondence

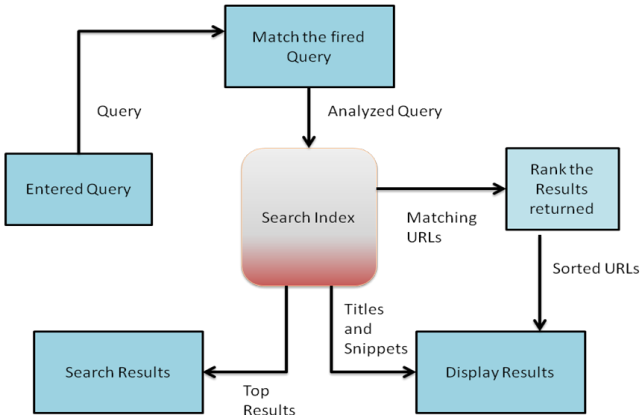

Figure 1. Mechanism of search engine.

results list. The relevance and popularity of web pages are generally computed by examining the hyperlink structure of web graph. The co-founders of Google i.e. Sergey Brin and Larry Page in 1998, developed a PageRank algorithm based on link analysis to calculate the prominence scores of web pages[2], which is thereafter used by Google as a part of its popular search engine.

The PageRank scores of web pages are computed from it's in links. PageRank algorithm is based on the principle that if a web page x is pointed by many other important web pages, then web page x is also important. As the Google becomes successful, the superiority of PageRank has been proved over other link analysis algorithms[3–5].

PageRank calculation is a challenging task. The first challenge is the extremely large volume of World Wide Web. Estimated size of World Wide Web is about trillions[6] of web pages. Therefore, a lot of computing efforts are required. And the second challenge results from the dynamic nature of World Wide Web. Every day new pages are added or removed from the WWW, thus pages and their content are continually changing in WWW. This results in a change in web structure[4]. Despite these challenges, the PageRank values must be state of art all the times and hence computation of PageRank scores must be performed as fast as possible. Looking at these challenges, it becomes vital to run the PageRank algorithm on High-Performance Machines. The search also returns a lot of spam pages to which the original PageRank is blinded[7]. The proposed algorithm assigns a non-uniform distribution of PageRank values to Web Pages, that is giving more importance to more important web pages and less importance to less important web pages.

At the present time, mainly three categories of web spam exist: The first one is link spam, second is content spam and the third category of web spam is cloaking: Link spam creates links between web pages to increase the ranking of spammer's websites. Content spam alters the contents of web pages by stuffing extra keywords associated with famous query terms. Cloaking indicates the way of presenting altogether different content to the user's browser as compared to the content presented to search engine spider[8,9].

The organization of this paper is specified as follows: Section 2 introduces related work done in the field of PageRank computation; Section 3 gives description related to GPU and CUDA; Section 4 discusses Traditional Page Rank Algorithm; Section 5 discusses the methodology used to parallelize Page Rank Algorithm using Non-Uniform PageRank Values; Section 6 describes experimental work and results in detail; Section 7 epitomize the proposed research work in the form of conclusion and future prospect. At last, references are presented.

2. Related Work

Many types of research have been done on ranking web pages. In[10] put forward a method to calculate review relevance by considering not only the similarity and correlation but also the votes for each review and proved that their method works well in terms of effectiveness and accuracy. In[11] proposed Link Click-Concept based Ranking Algorithm based on user profile construction. In[12] proposed a method to monitor the Simple Mail Transfer Protocol (SMTP) sessions and email addresses for detecting spamming messages. There has been an earnest effort to reduce the computational time of PageRank algorithm. In[13] presented Parallel Relaxed and Extrapolated algorithms based on the Power method for accelerating the PageRank computation. In[14] proposed PageRank computation in a distributed environment using iterative aggregation and disaggregation technique to effectively speed up the PageRank computation. In[15] presented a parallel computation of PageRank on a cluster of Gigabit PC and showed major improvement. In[16] applied Gauß-Seidel iterative method in a parallel computing environment for improving convergence. In[17] uses partition and repartition site based methods to balance load and minimize the communication overhead for parallel computations of PageRank values. In[18] perform experiments on a cluster of 70 nodes and evaluated that the PageRank computation on linear systems is agile as compared to power method. In[19] concentrated on

comparing the running time of PageRank algorithm on both GPU and CPU clusters. Other research work in this regard is by [20] who implemented power method for parallel computation of PageRank algorithm on AMD GPUs by using OpenCL programming language. Our research work is an attempt to reduce the PageRank values of spam pages so that important web pages are shown on the top of google search engine. We have implemented this work on NVIDIA Quadro 2000 GPU architecture using CUDA programming language.

3. GPU and CUDA

GPUs are processors conventionally designed for graphic processing in computer games. It is a multicore processor possessing extremely parallel processing ability with high-speed memories. GP-GPU exemplifies general purpose computing on graphics processing units to use GPU for computational jobs that are typically handled by CPU[3]. The architecture of GPU is shown in Figure 2. The GPU evolved due to the voracious market requisite for high quality and real time graphics. GPU, today is a processor with a plenitude of cores, which support high parallelism and multithreading with high memory bandwidth. The GPU runs on the SIMD model which is Single Instruction Multiple Data. Each core within a multiprocessor executes the identical instruction, but on dissimilar data. GPU contains one or more scalable multithreaded Streaming Multiprocessors (SMs). The SMs are designed to execute plenitude of threads concurrently.

CUDA (Compute Unified Device Architecture) allows programmers to define kernels that will execute on the GPU. A programmer is unaware of the hardware of the GPU instead they see the number of threads which are organized into blocks. The GPU's using CUDA can be utilized for General Purpose Computing. Unlike the CPU, the GPU has a parallel throughput architecture which helps in executing a plethora of threads concurrently[21,22].

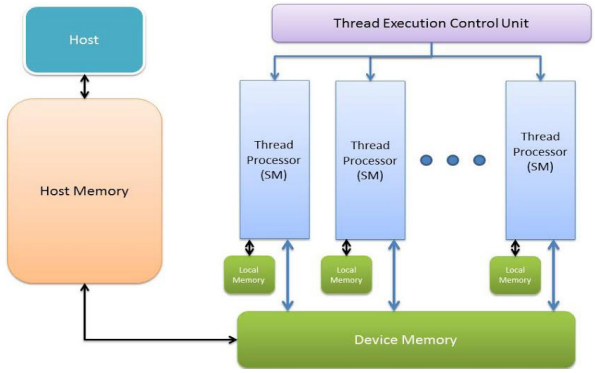

Figure 2. Architecture of GPU.

The grid is assigned to a GPU for execution, where each grid is further divided into blocks. A block is assigned to a multiprocessor (SM) and all the threads present within a block will make use of the processing elements of only its assigned multiprocessor. If we consider the case, where the number of SMs is less as compared to a number of blocks, then multiple blocks could be assigned to a single SM at a time, depending on the availability of memory in the SM. Threads present within all the blocks which are assigned to same SM will execute simultaneously. In an SM, threads present within a block are split in a group of 32 threads; such set of threads is called a warp. All threads of a warp execute in parallel and execute the same instruction at a time. An SM can execute 768 threads simultaneously, that is, 24 SIMD warps of 32 threads. Maximum parallelism can be achieved if all warps have 32 threads[21,22].

4. PageRank Algorithm

The famous search engine Google uses PageRank algorithm. This algorithm is based on assigning each web page a numeric weight, so as to measure the relative importance of each page within a hyperlinked structure of web pages. PageRank is based on the random surfer model[23] where a random surfer (a user) switches to a random page after clicking on several links. The value of PageRank determines the chance that the random surfer will arrive on that page by clicking on a link. This can be assumed analogous to a Markov Chain in which the pages are states and the equally probable transitions are links between the pages.

The Google's Algorithm as described in[2] is given as follows:

$$PR(A) = \frac{1-d}{N} + d \times \sum_{i} \frac{PR(B_i)}{\|B_i\|}$$

Where PR (A) is the PageRank of page A, PR (B_i) is the PageRank of pages B_i which links to page A,

B_i is the number of outbound links on page B_i and

d is a damping factor which can be set between 0 and 1.

The algorithm basically uses a uniform distribution of Page Ranks among the pages. The page which is pointed by several other pages becomes important. It does not consider the fact that some links are more important than

others and hence they should have more PageRank value as compared to other links. Furthermore, there is also a probability of raising spam web pages by the business firms, thereby enhancing hyperlinks to its home web page for the promotional and advertisement activities.

5. Improved Parallel PageRank using Non-Uniform Distribution

The proposed PageRank algorithm introduces a function *f(x)* in the calculation of PageRank scores of web pages. The function *f(x)* can be a polynomial, logarithmic or exponential function. The purpose of *f(x)* is to provide a non-uniform distribution among all the given PageRanks. The idea behind a non-uniform distribution comes from the fact that more important pages will have higher PageRank, they become more important as compared to less important pages and so for any single page having a link pointing to an important page should provide more amount of PageRank as compared to less important pages. The original PageRank[2,3] uses a uniform PageRank distribution for calculation of PageRank.

The formula used in our implementation is as follows:

$$PR(A) = \frac{1-d}{N} + d \times \sum_{i=1}^{m} \frac{f(PR(A))}{\sum_{j=1}^{\|B_i\|} f\left(PR(C_{ij})\right)} \times PR(B_i)$$

5.1 Proposed Parallel PageRank Pseudo Code

1 *G*: = set of pages.
2 **Threshold:** = convergence factor.
3 **function** f**(p)** // Non-uniform distribution function.
4 **for each** page *p* in *G* **do in parallel.**
5 *p*.prev_pageRank: = 1.0 // *p*.PageRank is the PageRank score of the page *p*.
6 **function** PageRank (*G*).
7 **for** step **from** 1 **to** k **do** // run the algorithm for k steps.
8 **for each** page *p* in *G* **do in parallel.**
9 *p*.pageRank: = 0;
10 **for each** page *q* in *p.incomingNeighbors* **do**
11 sum: = 0.0
12 **for each** page *r* in *q.outgoingNeighbors* **do**
13 sum += f(r.prev_PageRank)
14 *p*.PageRank += $\frac{f(p.prev_{pageRank})}{\textbf{sum}} \times q.prev_{pageRank}$
$\frac{f(p.prev_{pageRank})}{\textbf{sum}} \times q.prev_{pageRank}$
15 *p*.PageRank := (1-d) + *p*.PageRank * d // here d is damping factor
16 mx := 0 // to find maximum difference
17 **for each** page *p* in *G* **do** //find the maximum difference
18 mx := max of (*p*.PageRank - *p*.prev_PageRank, mx)
19 **if mx** is less than **threshold do** // Convergence reached
20 ***stop*** // stop the program
21 **for each** page *p* in *G* **do in parallel**
22 *p*.prev_PageRank : = *p*.PageRank

6. Experimental Results

The experiment aims to confirm that using a non-uniform distribution of PageRanks improves the PageRank of relevant pages with a reduction in the PageRank of irrelevant pages which are usually spam links. This method eliminates spam by giving more PageRank values to more important pages, whereas the spam pages which have a relatively low PageRank values are given an even lesser value than other pages. Hence the value of the spam pages will be suppressed. The proposed work uses different functions for calculation of PageRank values for non-uniform distribution, the functions used are: $f(x) = e^x$ -1, Logarithmic Function: $f(x) = \log_2(x+2) - 1$ and Square Function: $f(x) = x^2$.

We have implemented both the existing PageRank Algorithm and Proposed PageRank Algorithm on NVIDIA Quadro 2000 GPU architecture using CUDA programming language. The PageRank values obtained from both the existing and proposed PageRank Algorithm are then compared shown in the Tables 2, 3 and 4.

We have used CUDA on 64 bit Windows platform for the parallel implementation of PageRank. The CPU has an Intel Xeon processor clocked at 2.30 GHz with 4 GB of RAM. The GPU used is NVIDIA Tesla C2075 with compute capability 2.0.

The experiment is performed on three datasets taken from Stanford Large Network Dataset Collection[24]. These

are the Amazon's Product Co-Purchasing Network, Internet Peer to Peer Network Web graphs. The details of datasets are shown in Table 1. The input data were sorted according to the in-degree of the graph for better performance.

Dataset Amazon's Product Co-Purchasing Network -"Amazon302" consists of 26211 nodes and 1,234,877 edges. Only a few nodes are shown in Table 2 to visualize the results properly. It can be seen from the Table 2 how the values for each node changes according to the function. The node number 10 shows an increment in PageRank value in this case. Only the relevant or important pages will show a high-valued local maximum in Table 2. Other nodes show

Table 1. Properties of the data sets[20] used in the experiments

Dataset Name	Type	Nodes	Edges	Description
Amazon0302	Directed	262,111	1,234,877	Amazon product co-purchasing network from March 2 2003
p2p-Gnutella31	Directed	62,586	147,892	Gnutella peer to peer network from August 31 2002
Web-NotreDame	Directed	325,729	1,497,134	Web graph of Notre Dame

Table 2. PageRank values of some nodes of dataset 1: Amazon's Product Co-Purchasing Network -"Amazon302"

Node No.	Parallel PageRank	Proposed Parallel Pagerank Algorithm		
		Exponential Function	Logarithmic Function	Square Function
4	0.187242542	0.150228	0.152202	0.150006
5	2.653685497	3.09225	2.56103	3.27858
6	32.39687142	22.0483	51.2055	21.9181
7	45.6258104	28.3825	87.9957	28.4655
8	49.54418857	0.533287	89.6108	0.5795
9	223.3873659	1884.57	617.616	2199.66
10	41.28655134	0.309471	71.9561	0.195143
11	13.17856673	6.20527	14.9291	2.23216
12	6.628396633	6.97554	7.36729	7.47488
13	6.768893738	27.6338	10.9088	29.7742
14	103.6555688	0.82321	294.436	2.72674

mitigation in PageRank values, these nodes are actually considered as spam or irrelevant pages.

The proposed algorithm is implemented on dataset "p2p-Gnutella31" which consists of 62,586 nodes and 147,892 edges. We have shown only a few nodes in table 3 for better visualization. A couple of local maxima can be seen in the Table 3 which correspond to relevant pages. The Pagerank values for relevant pages are enhanced by using Exponential, Log and Square functions, which can be seen from the Table 3. Nodes whose PageRank values are suppressed are considered to be spam pages.

Table 4 shows the implementation of existing proposed algorithm on dataset "web-NotreDame" which consists of 325,729 nodes and 1,497,134 edges. We have shown only

Table 3. PageRank values of some nodes of dataset 2: Gnutella Peer to Peer Network – "p2p-Gnutella31"

Node No.	Parallel PageRank	Proposed Parallel Pagerank Algorithm		
		Exponential Function	Logarithmic Function	Square Function
1	0.541431794	1.06606	1.052	1.05132
2	0.742006559	2.61749	2.08039	7.98779
3	0.349563626	0.267324	0.378557	0.163827
4	0.963083037	4.33025	2.60139	10.2946
5	0.252753979	0.175646	0.198987	0.150852
6	0.37433706	0.210706	0.292558	0.150535
7	0.654885792	1.93907	1.75213	3.09926
8	0.380693168	0.214274	0.290963	0.151892
9	0.41766649	0.299011	0.424158	0.166012

Table 4. PageRank values of some nodes of dataset 3: Web graph – "web-NotreDame"

Node No.	Parallel PageRank	Proposed Parallel Pagerank Algorithm		
		Exponential Function	Logarithmic Function	Square Function
51	2.75361792	0.398098	6.36853	0.398025
52	3.94637185	0.205065	23.901	0.156454
53	2.13233258	0.166166	6.52146	0.15188
54	5.26112282	1.17416	10.3334	1.3005
55	6.0156624	0.319871	10.1728	0.357065
56	42.2198826	6.81718	70.2482	12.1533
57	1.99740612	0.15	1.19805	0.15
58	2.22523246	0.15	1.33911	0.15
59	33.1424986	15.1747	64.4203	23.1498
60	12.2011788	0.213027	28.28	0.150216

some of the nodes in Table 4, this table clearly depicts how the PageRank values are enhanced for the relevant pages. The important pages still remained important but with greater Pagerank values. As seen from the table, proposed algorithm exemplifies higher peak values for the relevant web pages.

7. Conclusion

This paper proposed an improved PageRank algorithm based on the non-uniform distribution of PageRank scores to calculate the PageRank of web pages. The proposed work renders spam filtering by classifying important as well as irrelevant web pages. This filtering is based on the hyperlink structure of the web that is related to out-bound and inbound links to a webpage. The apparent importance of a web page is determined by its PageRank values. Important pages show higher values whereas the spam pages show lesser values. Further improvements to spam filtering can be done using the web content mining. We performed the experiments on CPU and GPU using CUDA programming language with different standard datasets.

8. References

1. Article title. 2015. Available from: http://www.infotoday.com/searcher/may01/liddy.htm
2. Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Computer Network and ISDN Systems. 1998; 30(1-7):107–17.
3. Duong NT, Nguyen QAP, Nguyen AT, Nguyen HD. Parallel PageRank computation using GPUs. Proceedings of the Third Symposium on Information and Communication Technology ACM; New York, USA. 2012 Aug. p. 223–30.
4. Dubey H, Roy BN. An improved PageRank algorithm based on optimized normalization technique. IJCSIT. 2011; 2(5):2183–8.
5. Tarun K, Parikshit S, Ankush M. Parallelization of PageRank on multicore processors. Distributed Computing and Internet Technology. Proceedings of 8th International Conference, ICDCIT 2012; Springer Berlin Heidelberg: Bhubaneswar, India. 2012 Feb; 7154:129–40.
6. Article title. 2015. Available from: https://www.google.co.in/insidesearch/howsearchworks/crawling-indexing.html
7. Pu BY, Huang TZ, Wen C. An improved PageRank algorithm: Immune to spam. IEEE Fourth International Conference on Network and System Security; Melbourne. 2010 Sep. p. 425–9.
8. Hua J, Huaxiang Z. Analysis on the content features and their correlation of web pages for spam detection. IEEE Transactions on Communications. China. 2015 Mar; 12(3):84–94.
9. Almomani A, Obeidat A, Alsaedi K, Obaida MAH, Al-Betar M. Spam E-mail filtering using ECOS algorithms. Indian Journal of Science and Technology. 2015 May; 8(S9). DOI:10.17485/ijst/2015/v8iS9/55320.
10. Shri JMR, Subramaniyaswamy V. An effective approach to rank reviews based on relevance by weighting method. Indian Journal of Science and Technology. 2015 Jun; 8(11). DOI: 10.17485/ijst/2015/v8i11/61768\.
11. Geetha Rani IS, Sorana Mageswari M. A link-click-concept based ranking algorithm for ranking search results. Indian Journal of Science and Technology. 2014 Jan; 7(10). DOI:10.17485/ijst/2014/v7i10/50682.
12. Anbazhagu UV, Praveen JS, Soundarapandian R, Manoharan N. Efficacious spam filtering and detection in social networks. Indian Journal of Science and Technology. 2014 Nov; 7(S7). DOI: 10.17485/ijst/2014/v7iS7/61956.
13. Arnal J, Migallon H, Migallon V, Palomino JA, Penades J. Parallel relaxed and extrapolated algorithms for computing PageRank. The Journal of Supercomputing. 2014 Nov; 70(2):637–48.
14. Zhu Y, Ye S, Li X. Distributed PageRank computation based on iterative aggregation-disaggregation methods. Proceedings of the 14th ACM International Conference on Information and Knowledge Management; New York, USA. 2005 Oct. p. 578–85.
15. Manaskasemsak B, Rungsawang A. Parallel PageRank computation on gigabit PC Cluster. Proceedings of 18th IEEE International Conference on Advanced Information Networking and Applications AINA. 2004 Mar; 1:273–7.
16. Kohlschiiutter C, Chirita PA, Nejdl W. Efficient parallel computation of PageRank. Advances in Information Retrieval. 28th European Conference on IR Research, ECIR 2006, Springer Berlin Heidelberg,; London, UK. 2006 Apr; 3936:241–52. April 2006. pp. 241–252.
17. Cevahir A, Aykanat C, Turk A, Barla Cambazoglu B. Site-based partitioning and repartitioning techniques for parallel PageRank computation. IEEE Transactions on Parallel and Distributed Systems. 2011 May; 22(5):786–802.
18. Gleich D, Zhukov L, Berkhin P. Fast parallel PageRank: A linear system approach. Technical Report; 2004.
19. Cevahir A, Aykanat C, Turk A, Barla Cambazoglu B, Nukada A, Matsuoka S. Efficient PageRank on GPU clusters. IPSJ SIG Technical Report. 2010; 2010(21):HPC-128.
20. Wu T, Wang B, Shan Y, Yan F, Wang Y, Xu N. Efficient PageRank and SpMV computation on AMD GPUs. IEEE International Conference on Parallel Processing; San Diego. 2010 Sep. p. 81–9.

21. Article title. 2015. Available from: http://docs.nvidia.com/cuda/cuda-c-programming-guide
22. Bhatia S, Tolpadi M, Rasool A. Importance of GPGPUs in efficiency improvement of real world applications. IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS); Bhopal. 2014 Mar. p. 1–6.
23. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing Order to the Web. Stanford University: Technical report; 1999.
24. Article title. 2015. Available from: https://snap.stanford.edu/data