IJMLNCE JOURNAL

International Journal of Machine Learning and Networked Collaborative Engineering

Journal Homepage: http://www.mlnce.net/home/index.html

DOI : https://doi.org/10.30991/IJMLNCE.2020v04i01.001

Efficient Distributed Web Crawler Using Hefty and Enhanced Bandwidth Algorithms for Drug Website Search

[a] Ramachandran, [b] R. Arunpraksh, [c] Aghila Rajagopal, [d] Manju Khari

[a]Assistant Professor, Department of CSE, University College of Engineering, Panruti, ramautpc@gmail.com*

[b]Assistant Professor, Department of CSE, University College of Engineering, Ariyalur, arunitvijay2014@gmai.com

[c]Associate Professor, Department of IT, Sethu Institute of Technology, Kariapatti.aghila25481@gmail.com

[d]Assistant Professor, Ambedkar Institute of Advanced Communication Technologies and Research,manjukhari@gmail.com,

ORCID:0000-0001-5395-5335

Abstract

Prefabricate a proficient search structure is very important due to the current scale of the web. Search engines mine information from the web and a program called a web crawler, which efficiently surfs the web. A distributed crawler belongs to a variant of a web crawler, uses a dispersed computation method. In this paper, we design and implement the concept of Efficient Distributed Web Crawler using enhanced bandwidth and hefty algorithms. Mostly Web Crawler doesn't have any distributed cluster performance system and any implemented algorithm. In this paper, a novel Hefty Algorithm and enhanced bandwidth algorithm are combined for a better-distributed crawling system. The hefty algorithm was implemented to provide efficient and robust surfing results while applying on the drug web search. We also implemented the Enhanced Bandwidth algorithm to improve the efficiency of the proposed crawler.

Keywords

Distributed crawler

Web crawler

Bandwidth

1. Introduction

A web crawler is a meta-search engine, which combines the top search results from the represented search engines. The search of users may also involve the audio, video, news, and more. A web crawler is a program or script, also called a web spider or web bot crawls the web automatically. A lot of trustworthy sites, particularly search engines, use the spider process to provide a summary of website content. The website content includes the Uniform Resource Locator (URL) and the text-based summary. The Web Crawlers generate a list of URLs of pages that provide information for later search, leads to the reduction of time and resources. Maintenance tasks such as checking links and validation of HTML codes are carried out by these web crawlers. Harvesting of e-mail addresses from web pages is done through this crawler.

The algorithm for the first web crawler, World Wide Wanderer, was designed in 1993, has not been updated since. Almost all crawlers go along with some modifications to the basic web-traversal algorithm. A Crawler is a massive search engine with the following issues. First, it should have an excellent crawling technique, i.e., which file should be downloaded next. Second, it must have a robust architecture, able to handle downloading of a large number of pages per second in spite of crashes, considerate and manageable

resources in web servers. Web crawlers also take into account scalability because of the growing size of the World Wide Web. The acceleration of traversing web can be limited by factors such as latency and bandwidth of the networks. Determining the modification of a web page, will helps to minimize the level of unwanted polling done by a crawler. The polling will exploit a minimal amount of resources, the more that can be spared to the task of locating new information. Finally, crawlers depend upon one another by communicating with each other and being occurrences of themselves (in the parallel sense). This ascends the need for unconventionally cooperative sharing web crawlers. When there is a need, crawlers make decisions and communicate with each other.

Distributed web crawling is a distributed computing technique on the web where many crawlers are working to scatter the web crawling process, which establishes the maximum exposure to the web. A median proxy manages the exchanging information and synchronization of the nodes, as it is geographically scattered as the World Wide Web. The page rank algorithm is generally used to raise the efficiency and quality search from the web. This Page ranking for a web sometimes has a low rank, i.e., 2 or 3 or sometimes 0. The high-ranking page will have enormous traffic, and the overall performance is not good enough. And also, the surfing results occur only at the time of indexing and not at the query time. In the distributed fashion of web crawlers, multiple crawlers receive individual URL from a URL server. These crawlers then start downloading the web pages simultaneously. The central indexer receives the downloaded pages from the crawlers, on which links are mined and sent through the URL server to the crawlers. Implementation of web crawlers in a distributed fashion using a cluster system is a difficult task because of its enormous resource usage.

Apart from the page rank algorithm, numerous distributed crawling algorithms crawl for the web pages from the web server in parallel in a coordinated fashion. The coordination technique improves efficiency by avoiding repeated crawling on similar documents. Without coordination, the crawler will individually explore its queue without considering the other crawler's queue since this crawling without coordination will not be infrequent. So, the crawling algorithm should have the coordination techniques supported by link analysis metrics, which are strong enough due to the initial seed documents, i.e., the cluster of records are fetched from unlike seeds are mostly overlapping. Without coordination, this robustness leads to the parallelization effect of crawling the same documents.

The section I involves the system architecture; section II includes the proposed model of the system, section III consists of the performance analysis.

2. Related Work

In paper [15], Web crawling uses the map-reduce programming paradigm and some features of cloud computing. The crawler crawls the web by using disseminated agents, and each agent stores its discovery on a Cloud Azure Table. But it takes more time to access the unstructured data. In another paper, the Topic Sensitive Page Ranking algorithm [16] has been used; the search process is carried out in context by highlighting the words on a web page. At query time, these importance scores are combined together based on the subject of the query to form an integrated PageRank score for those pages matching the query. This uses three methodologies for scoring the pages they are query sensitive scoring, Query Time Importance Score, and topic sensitive scoring.

In paper [11], commonly used link analysis algorithm such as page rank, weighted page rank, and HITS (Hyperlink induced topic search) was discussed. The drawback of the page rank algorithm is query independent, and results are sorted according to the importance of pages. The weighted page rank algorithm is also query independent and consumes more indexing time. The HITS have the topic drift and efficiency problem. In paper [12], propose a method to evaluate the relevance of a page to a searched query-based not only on the information contained in its textual content but also by finding the relevance of the linked pages to the current page w.r.t. to the given searched criteria. The proposed algorithm represents every page by a vector of terms using the Vector Space Model technique (VSM). The relevance of each time is estimated by the Vector Space Model technique utilizing the support of occurrence information, which assigns weights for every term in a document and represents the documents as term frequency weight vectors. This is a time consuming process to generate the weights of the documents.

In paper [13], from the inferred similarity graph visually, "authority" nodes were identified by the

tasks of image ranking algorithms and proposed Visual Rank to analyze the visual link structures among images. The images identified as "authorities" are selected to answer the image queries well. Our experimental reveals show significant improvement, for user satisfaction and relevancy, in comparison to recent Google Image Search results. Maintaining modest computational cost is vital to ensuring that this procedure can be used in practice; however, this requires more resource space to find the large-scale computing process.

In paper [17], Jaytrilok Choudhary proposed the priority-based semantic web crawler. The semantic score is determined for the given URL, which is obtained from the web page. The semantic score is intended from the anchor text of the unvisited URL. By using the semantic score of the pages, the web pages are getting prioritized. Topic ontology can also be used to determine the semantic score. The scoring is only focused on the semantic, not on the similarity of the content of the web page and also not having the experimental outcomes. In paper 19, a parallel migrating crawler is dividing the crawling work to various independent and similar crawlers, which moves to multiple machines to improve network efficiency and reduce the time for downloading. Experiments and results of proposed migrating and nature of the parallel working of the design were recorded. It has the central database communicates with the search engine. The central database stores the URLs received from the application, and it also stores the final downloaded documents which are retrieved by various crawlers since this uses the identical old mapper system to map the request and response between the search engine and the crawler.

In paper [7], Vladislav Shkapenyuk et al., the high-performance distributed web crawler by using the crawler application. The crawling application adopts what page to demand next given the current state and the formerly crawled pages, and issues a stream of requests (URLs) to the crawling system. The crawling system copies the requested pages and passes them to the crawling application for investigation and storage. The crawling system is responsible for tasks such as robot barring, speed control, and DNS resolution that are common to most scenarios, while the application implements crawling strategies such as breadth-first or focused. The crawling application replicates itself while there is a need for large web pages leads to poor scalability.

3. Existing Work

From the related work, we have identified a few problems in the distributed web crawling.

- In web crawling, network traffic is a big issue while fetching a vast amount of web pages.
- Load balancing is the main issue, while fetching a vast amount of web pages. However, there are many proposed algorithms to improve scalability, crawling speed, URL prioritization. Only some of them have been implemented.
- Bandwidth utilization is a critical issue for distributed crawling.
- So far, deep websites are utilized to search for drug details. Alphabay, Valhalla, Hansa, Zocalo Marketplace are the currently available deep websites.

4. Design Objectives

- **Selecting URL and Content:** A web crawling is desirable to have some mechanism to rank URL/domains to provide good seed and content selection.
- **Freshness:** Some websites are to be frequently updated leads to updating in the crawling content also. It should maintain a strategy to identify the schedule for every link to be downloaded.
- **Deep Crawling: C**rawling till to the lower level of the web, to identify the unvisited websites and list out those in the URL list.
- **Focussed Crawling:** It is the estimation of the unvisited websites that are near similar to the content before downloading the web page. So, there is a periodical update of the whitelist.
- **Challengers:** Every web crawler will face some irrelevant processes such as crawling traps, spam sites, cloaked content. The crawling trap is the collectives web site that creates uncountable sets of URLs for the crawler to find. The spam sites are malicious web sites that exploit user's resources

and bandwidth. Cloaked content is to serving a web page for search engines and providing entirely different ones to the user.

- **Politeness:** The site operator bans the misbehaving crawlers. To avoid this condition, tracing the robots.txt on every site to follow their terms and condition.

5. Architecture

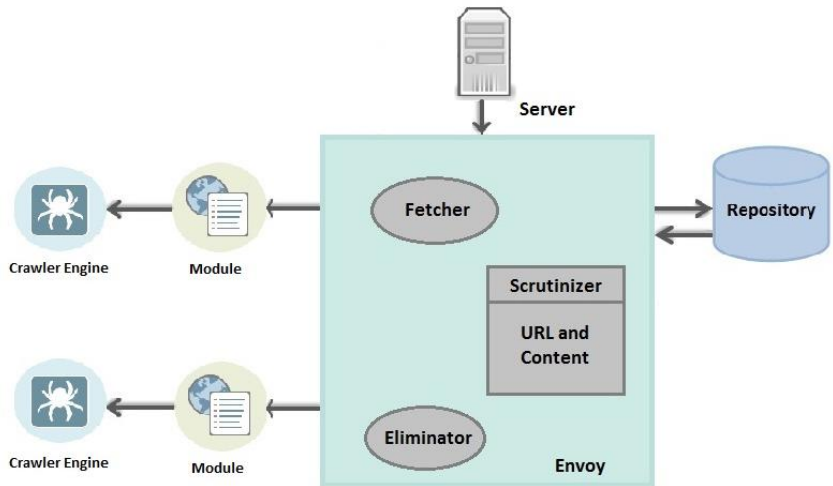

Figure 1: The System Architecture

The main segments in the architecture are the Crawler engine, Module, Envoy, server, and repository. The crawler engine has an automated script to mine the web, where it simultaneously uses several search engines for searching the web (such as google, yahoo, bing). The crawler engine searches the web to collect clusters of URLs typically denoted as seed URL documents. Additionally, the crawlers validate the HTML code, hyperlinks, and web scraping. At starting, the keyword is entered into the web crawler engine, and it sends the request to the webserver. The web server is the computer system that serves the user request.

The envoy is a crawling agent that consists of the following phases, such as fetcher, Scrutinizer, and Eliminator. The fetcher fetches the URL from the web server with the most related terms. The Scrutinizer scrutinizes the URL and then content to give priority in the more relevance URL to the keyword. The eliminator will remove the visited URL request from the list and replace it with the unvisited URL. Then the URL list is extracted and denoted as the modules and displayed in the crawling web engine. The repository saves the URL lists that are crawled from the webserver.

6. Proposed Model

In our design, the hefty algorithm and bandwidth algorithm are combined to give an efficient crawling strategy. Initially, the user mentions the keyword in the crawler engine; the list of URLs is extracted from the webserver. The collected URL is partitioned into separate sets, here noted as URL modules. Each URL module is handled by a different web crawler. Each Web Crawler parses and logs into the URL that is present inside the module of it, and the remaining URL is sent to the corresponding module. Each crawler has prior knowledge of the search table relating to the URL modules (maps IP address) by identifying the crawler threads.

The first URL in the module is surfed, and then the next URL will be in the queue for the next visit.

Here the envoy is an agent where the modules have been processed. The identified URL has been analyzed by comparing it with the other URL in the module. If the same visited URL is found, then the URL has been eliminated from the module. A new unvisited URL will be added to the module. After the URL has been surfed the content inside in it was scrutinized. If it has similar content (which is already surfed), then that URL has also been replaced with a new one.

The crawler is composed of a collection of fetchers, present inside the envoy. Each element is run as a distinct thread, and the pool of fetcher threads are shaped upon startup. A fetcher pauses until the envoy triggers it for salvaging a given URL. Then it makes an HTTP request to the suitable Web server and, after having moved the document, it yields to a wait state. Fetchers are fed to implement the control policy for the crawlers. The manager selects the URLs to be recovered during their primacies, and it activates the optimum number of fetchers to deed the assigned bandwidth. The duplicate detector is used to identify the duplicate URL or similar URL to avoid the excessive usage of memory for the repository. This will improve search engine quality.

Distributed Crawling-Hefty Algorithm (DC-HA):

The distributed crawling algorithm is categorized into three stages, such as the starting stage, formation stage, and relocation stage. In the starting stage, the keyword is entered, which extracts a seed set of documents for various crawlers. In the formation stage, the different URLs are organized by using their priority. The similar URL and the visited URL are identified to filter out. The estimation of all related information is used to find which documents to fetch next. This process carried out while retrieving the current document itself—an interaction between crawlers in order to exchange already analyzed documents called the relocation stage.

The duplication in the URL can be easily identified because it will be completely identical, whereas the near duplicate has some structural difference. Few structural changes are date change, small edits, metadata, and other changes. The similarity score estimates the close duplication. The resemblance between the URL is computed to identify the near-duplicates between the web pages. The similarity score counts the similarity degree between the web pages. The higher computation score indicates a higher percentage of similarity. So, the higher computation value suggests the web page have more near duplicates. The computation value estimates all the similar pages whose similarities are above the given threshold. The similarity value is between the two webpages is calculated as follows:

The keywords for the web pages and the counts of the keywords are represented as

$P_1=\{(K_1,C_1),(K_2,C_2),(K_3,C_3),\ldots\ldots,(K_n,C_n)\}$

$P_2=\{(K_1,C_1),(K_2,C_2),(K_3,C_3),\ldots\ldots,(K_n,C_n)\}$

At first, the similarity between the web pages is measured by using the keyword for the first web page P_1. This is calculated by obtaining the difference between the number of occurrences of both the keywords. If the keyword is not on the web page means then the computation value is zero.

$$C_v[P_1]=\frac{1}{N1}\sum_{i=1}^{N1} Abs[P1(C,Ki)-P2(C,Ki)] \rightarrow [1]$$

if K_i does not belong to P_2, then count=0

Where $N_1= | P_1|$.

Then the remaining keywords (R_K) have been taken and find the similarity measure for those keywords in another web page P_2.

$R=P_2-P_1$

$$C_v[P_2]=\frac{1}{N2}\sum_{i=1}^{N2} Abs\ R[C,Ki] \rightarrow [2]$$

Where $N_2=|R|$.

Gradually, the final similarity score is calculated as follows:

$$C_V[K_i]= C_V[P_1]+ C_V[P_2] \rightarrow [3]$$

The similarity between the two pages is obtained by the computing value $C_V[K_i]$. The near duplicates of the web pages are obtained by using the data present in the repository. The web page with the highest computation value is deleted from the module, i.e., URL list

Algorithm 1: DC-HA

Input: Enter the Query

Output: List of URL

1. Start
    a. Enter the query (say Q).
    b. Obtain URL list U_i.
    c. Visit the first URL U_1 //by First In First Out.
    d. Scrutinize $S(U_i)$
    e. Compare the URL U_1 with the next URL U_2
    f. Calculate the similar computation value
        1. If the computation score is higher
            a. Remove URL
            b. Add $U_{new}=U_{j+1}$
        2. else
            a. Repeat $S(U_i)$
        3. End if
            a. Scrutinize $C(U_i)$
            b. Remove U_i+1
            c. Add $U_{new}=U_{j+1}$

4. else

    a. Repeat $S(U_i)$

5End if

End

Bandwidth Algorithm (BW-A):

To select the next URL U, the estimation of the bandwidth BW(U) by the envoy, and the consumption of retrieving the file (f) leads to the prediction of a fetcher. After downloading the file, the envoy can measure the actual rate of transfer as E(U) and also used to calculate the server speed until the end of the process. The envoy can compute the predicted total rate of transfer of the crawler DC_{BW} as u∈S. BW(F), where F is the group of the file that is currently given to the fetchers. By presumptuous the crawler which previously starts the fetcher, after the fetcher finishes its current download U_c, the freed bandwidth will be allotted to one or more downloads from starting. Hence, the crawler bandwidth is updated as $C_{BW}=C_{BW}-BW(Uc)$ and the envoy looking for document U_f in the queue, when downloading it, estimated bandwidth will be under the level of assigned threshold L, i.e., $C_{BW} + BW(U_f) \leq L$. The URL search begins from the first link in the queue and makes up to a given maximum depth max_d, by taking into account of high priority documents. Envoy begins to activate a fetcher for downloading U_f, if not , it stops and starts to wait for another download completion. When the new download does not use the available slot to the expected bandwidth, the queue is searched for a further reference link.

Algorithm 2: BW-A

1. Set f=0
2. Retrieve the file U_f at depth d in the URL queue
3. If $C_{BW} + BW(U_f) \leq L$, assign U_f to an inactive fetcher and update $C_{BW} = C_{BW} + BW(U_f)$
4. Set f = f +1
5. Repeat step 2 until $C_{BW} \leq L$ or f=max_d

7. Experimental Work

The NetBeans IDE 8.1 has been used to experiment the entire system. The web crawler is implemented using the single system to measure the performance of pages crawled and the bandwidth of the node. After several testing and debugging, these data are obtained. The web crawler (single node) has 2GB RAM, Core2Duo 2.8 GHz Processor and the bandwidth limit was set to 1.5 Mbps for crawling the data. Initially for the given keyword 40 URL's are crawled and denoted as the seed URL. The measurement of bandwidth utilization is recorded for 1000 minutes.

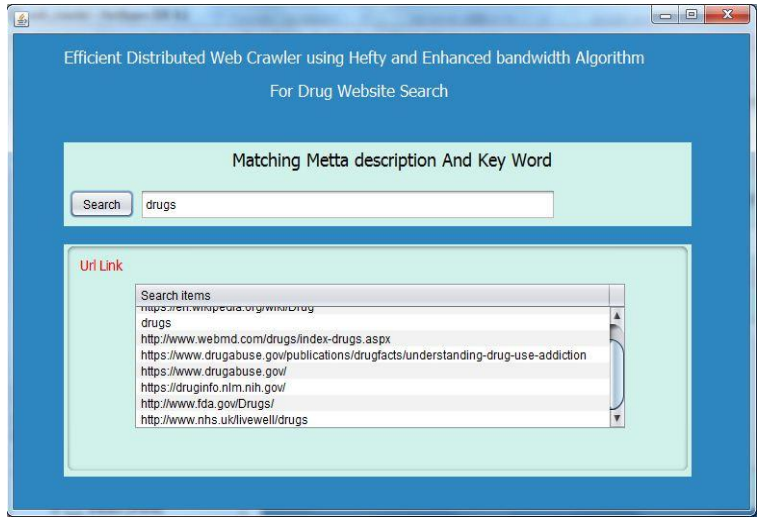

Figure 2: Represents the initial searching process carried out using the DC-HA

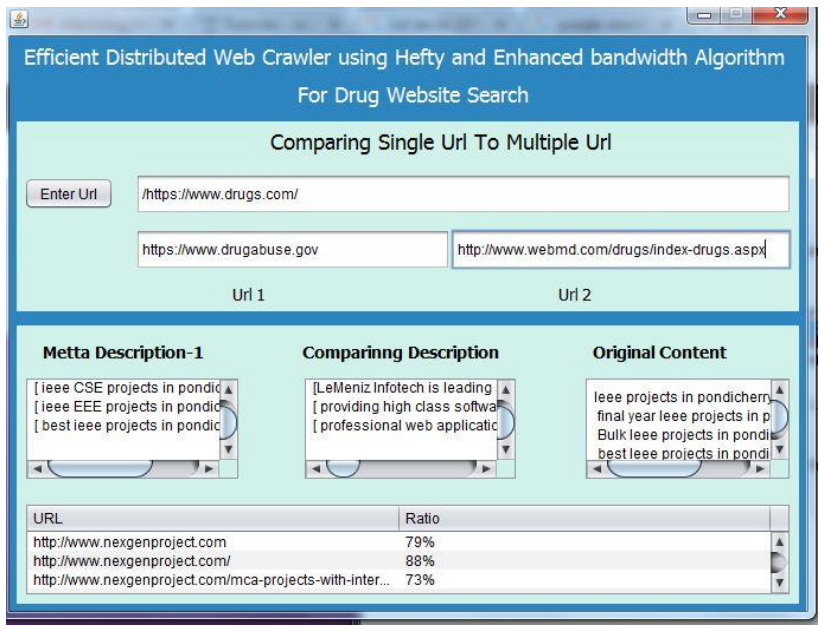

Figure 3: Comparing the first URL with the next URL

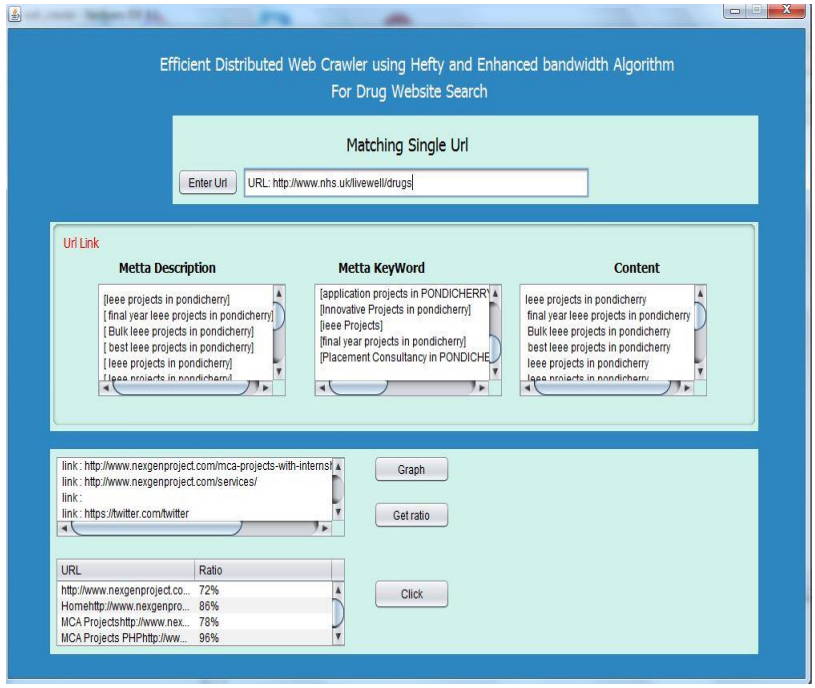

Figure 4: Comparing the initial URL content with the upcoming URL

8. Conclusion

Generally, search engines compete against each other by size and currency of the underlying database, also consider the quality and response time of their ranking function. Distributed web crawler architecture was introduced for crawling web pages and provided that index service. In this paper, the combination of hefty algorithm and bandwidth algorithm features provides an excellent outcome. The architectural design has given the tasks of Web crawler components. The experimental work shows the improved number of web pages downloaded and operates within the limited bandwidth using drug websites.

References

[1]. S. Saranya, B.S.E. Zoraida and P. Victor Paul, "A Study on Competent Crawling Algorithm (CCA) for Web Search to Enhance Efficiency of Information Retrieval", Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, 2015.

[2]. P. Jaganathan and T. Karthikeyan, "Highly Efficient Architecture for Scalable Focused Crawling Using Incremental Parallel Web Crawler", Journal of Computer Science 2015, vol. 11 (1): 120.126.

[3]. Subhendu kumarpani, Deepak Mohapatra, Bikram Keshari Ratha, "Integration of Web mining and web crawler: Relevance and State of Art", International Journal on Computer Science and Engineering, Vol. 02, No. 03, 2010, 772-776.

[4]. Monica Peshave, and Kamyar Dezhgosha, "How Search Engines Work And A Web Crawler Application".

[5]. Taekyoung Kwon and Yanghee Choi, Sajal K. Das, "Bandwidth Adaption Algorithms for Adaptive Multimedia Services in Mobile Cellular Networks".

[6]. Raja Iswary, and KeshabNath, "Web Crawler", sInternational Journal of Advanced Research in Computer and Communication Engineering,Vol. 2, Issue 10, October 2013.

[7]. VladislavShkapenyukTorstenSuel, "Design and Implementation of a High Performance Distributed Web Crawler", NSF CAREER Award NSF CCR-0093400, Intel Corporation, and the New York State Center for Advanced Technology in Telecommunications (CATT).

[8]. Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D.Weitz. "Approximating aggregate queries about web pages via random walks". In Proc. of 26th Int. Conf. on Very Large Data Bases, September 2000.

[9]. M. Najork. Atrax: "A distributed web crawler", Presentation given at AT&T Research, March 20, 2001.

[10]. Zhixing GAO, Kunhui LIN, "Design and Implementation of a High Performance Distributed Web Crawler", Journal of Computational Information Systems5:6(2009) 1817-1823.

[11]. Rekha Jain, Dr. G. N. Purohit "Page Ranking Algorithms for Web Mining", International Journal of Computer Applications (0975–8887) Volume 13–No.5, January 2011.

[12]. P. C. Saxena, J. P. Gupta, Namita Gupta, "Web Page Ranking Based on Text Content of Linked Pages", International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010.

[13]. Yushi Jing, and Shumeet Baluja, "Visual Rank: Applying PageRank to Large-Scale Image Search", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 11, November 2008.

[14]. T. Haveliwala, "Topic-Sensitive Page rank: A Context-Sensitive Ranking Algorithm for Web Search," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 784-796, July/Aug. 2003.

[15]. Mehdi Bahrami, Mukesh Singhal, Zixuan Zhuang, "A cloud-based web crawler architecture", Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on 17-19 Feb. 2015.

[16]. Taher H. Haveliwala, "Topic Sensitive Page Rank", Supported by NSF Grant IIS-0085896 and an NSF Graduate Research Fellowship. May 7–11, 2002.

[17]. Jaytrilok Choudhary, Devshri Roy, "Priority based Semantic Web Crawler", International Journal of Computer Applications (0975 –8887), nov-2013.

[18]. Mohd Adil Siddiqui, Sudheer Kumar Singh, "URL Ordering based Performance Evaluation of Web Crawler", International Journal of Computer and Information Technology (ISSN: 2279 – 0764), jan-015.

[19]. Akansha Singh, Krishna Kant Singh, "Faster and Efficient Web Crawling with Parallel Migrating Web Crawler", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 11, May 2010.

[20]. Ali Selamat,Fatemeh Ahmadi Abkenari, "Application of clickstream analysis as Web page importance metric in parallel crawlers", International Symposium on Information Technology (Volume:1), 15-17 June 2010.

[21]. Bing Zhou, Bo Xiao, Zhiging Lin, Chuang Zhang, "A distributed vertical crawler using crawling-period based strategy", Future Computer and Communication (ICFCC), 2nd International Conference on (Volume: 1), 21-24 May 2010.

[22]. Nagappan V. K, P. Elango, "Agent based weighted page ranking algorithm for Web content information retrieval", Computing and Communications Technologies (ICCCT), International Conference on 26-27 Feb. 2015.

[23]. R. Khanchana; M. Punithavalli, "An efficient web page prediction based on access time-length and frequency", Electronics Computer Technology (ICECT), 3rd International Conference on 2011, Volume: 5

Author's Biography

A. Ramachandran is working as an Assistant Professor in the Department of Computer Science and Engineering, University College of Engineering Panruti, Tamilnadu, India. He completed his Master of Engineering at Annamalai University and pursuing a Ph.D. in Anna University. His areas of interest are Software Engineering, Data Mining, and Web Crawlers.

Dr. R. Arun Prakash has 14 Years of teaching experience in reputed institutions and universities. He received B.Tech. Degree in Information Technology from Bharathidasan University, Trichy, in 2003, M.Tech. Degree in Information Technology at Sathyabama University, Chennai in 2005, and a Ph.D. degree in the Department of Computer Science and Engineering at Anna University Chennai 2016. At present, he is an Assistant Professor in the Department of Computer science and Engineering, University College of Engineering, Ariyalur, Anna University, Tamilnadu, India. He is a lifetime member of ISTE. He has been a lecturer at the graduate and post-graduate level and has participated in several International and National level conferences and workshops. He has published around 18 papers in the reputed international journals and more than ten articles in the global and national conferences and contributed books in programming in C and two book chapters. His primary interest is currently M-commerce, Mobile computing, and image processing, and wireless networks.

Dr. Aghila Rajagopal, Associate Professor of Department of Information Technology, Sethu Institute of Technology, has more than 15 years of teaching experience. She received her Ph.D. degree from Anna University, Chennai, in 2015. Her research area includes Distributed Computing, Software Engineering. She has published many papers in National and International journals. She is a reviewer in various national, international Journals. She holds the membership in many professional bodies like CSI, IAENG, IEDRC, IACSIT

Dr. Manju Khari is an Assistant Professor in the Ambedkar Institute of Advanced Communication Technology and Research, Under Govt. Of NCT Delhi affiliated with Guru Gobind Singh Indraprastha University, Delhi, India. She is also the Professor- In-charge of the IT Services of the Institute and has experience of more than twelve years in Network Planning & Management. She holds a Ph.D. in Computer Science & Engineering from National Institute of Technology Patna. She received her master's degree in Information Security from Ambedkar Institute of Advanced Communication Technology and Research, formally this institute is known as Ambedkar Institute of Technology affiliated with Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are software testing, software quality, software metrics, information security, optimization, Artificial Intelligence, and nature-inspired algorithms. She has published more than 70 papers in refereed National/International Journals & Conferences (viz. IEEE, ACM, Springer, Inderscience, and Elsevier). She associated with many International research organizations as Editor of Springer, Wiley and Elsevier books and Guest editor of International Journal of Advanced Intelligence paradigms, reviewer for International Journal of Forensic Engineering, Inder Science, and editorial board member of International Journal of Software Engineering and Knowledge Engineering, World Scientific. Computer Science and Engineering of AIACT&R, Geeta colony, Delhi, India