Web Crawling by Means of Multithreading and Google Numerical Weighting Technique

Ms. Amrita Banjare[1] Mr. Rohit Miri[2] Miss. Khushboo Sharma[3]
[1,2,3]Dr. C.V. Raman University, Bilaspur

Abstract— A web crawler (also known as a spider or a robot) is an organism for the volume downloading of web pages. Web spidering may emerge to be simply an application of BFS (Breadth First Search) technique, the genuineness is that there are numerous challenges ranging from systems concerns like organizing very large data structures. For such a massive data structures, it became a substantial challenge for single process crawlers. Web crawlers are meant for various purposes. Most importantly, they are one of the main components of search engines, SEO (Search Engine Optimization). Henceforth compelling algorithms are in demand for efficient web crawling. As a consequence it has become very important to make effectual crawling procedure, so as to finish crawling process in a prudent amount of time. There are a lot of programs out there for web crawling but it required a Web Crawler that allowed trouble-free customization. In this paper we have proposed an effectual crawling mechanism in which integration of multithreaded crawler and Google Numerical weighting technique has been done. Numerical weight of webpage is a "vote" by all other pages on the web. By applying PR (Page Rank) it should bring high quality documents so that the user gets the required pertinent information within satisfactory time.

Key words: WWW, Bot, Spider, PR, BFS

I. INTRODUCTION

Web crawler is a web service that assists users in their web navigation by automating the task of link traversal, creating a searchable index of the web, and fulfilling searchers' queries from the index. That is, a web crawler automatically discovers and collects resources in an orderly fashion from the internet according to the user requirements. Different researchers and programmers use different terms to refer to the web crawlers like aggregators, agents and intelligent agents, spiders, due to the analogy of how spiders and crawlers traverses through the networks, or the term (robots) where the web crawlers traverses the web using automated manner.

Internet would have not become so popular if search engines would not have been developed. Starting in 1994, a number of search engines were launched, including AltaVista, Excite, Infoseek, Inktomi, Lycos, and of course the evergreen, Yahoo and Google. Most of these search engines save a copy of the web pages in their central repository and then make appropriate indexes of them for later search/retrieval of information. User interface, Query engine, Indexer, Crawlers and Repository are the basic components of search engines.

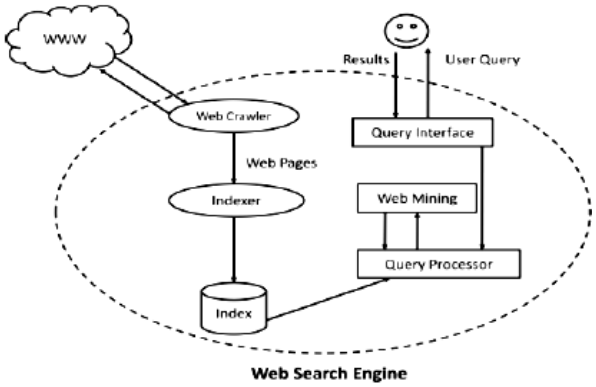

Fig. 1: Working of Search Engine

The remainder of this paper is organized as follows: Section 2 provides a brief review of the web crawlers and page rank and illustrates literature survey. In Section 3, we drew the attention towards problems in existing systems. In section 4 finally, we concluded our work.

II. BACKGROUND

A crawler also popularly known as spider or robot is a program which visits Web servers spread across the world irrespective of their geographical locations, downloads and stores Web documents on a local machine mostly on behalf of a Web Search Engine. A web crawler is a program that retrieves and stores web pages from the Web. A web crawler starts off by placing an initial set of URLs in a seed queue. The web crawler gets a URL from the seed queue, downloads the web page, extracts any URLs in the downloaded page, puts the new URLs in the seed queue, and gets the next URL from the seed queue. The web crawler repeats this crawling process until it decides to stop.

A. Page Rank:

PageRank[8] is a link analysis algorithm to measure the page relevance in a hyperlinked set of documents, such as the World Wide Web. This algorithm assigns a numerical weight to each document. This numerical weight is also called PageRank of the document. The PageRank of a web page represents the likelihood that a person randomly clicking will arrive at this page. The PageRank algorithms requires several iterations to be executed. At each iteration, the values will be better approximated to the real value. In its simplest form, PageRank uses the next formula for each web page at each iteration:

$$PR(u) = \sum_{v \epsilon Bu} PR(v)/L(v)$$

In Topic Sensitive PageRank [4], several scores are computed: multiple importance scores for each page under several topics that form a composite PageRank score for those pages matching the query. During the offline crawling process, topic-sensitive PageRank vectors are generated,

using as a guideline the top-level category from Open Directory Project (ODP). At query time, the similarity of the query is compared to each of these vectors or topics; and subsequently, instead of using a single global ranking vector, the linear combination of the topic-sensitive vectors is weighed using the similarity of the query to the topics. This method yields a very accurate set of results relevant to the context of the particular query. With Topic Sensitive PageRank a set of ranking vectors are computed, as opposed to the single PageRank vector generated using standard PageRank. These vectors are biased using a set of representative topics, to capture the notion of importance with respect to a topic, indirectly specified through a user query and if available through user context also. Some of the related survey papers have been illustrated in the following table:-

S. No.	Author	Title	Findings
1.	A. Heydon and M. Najork. Mercator	"A scalable, extensible web crawler."	The work in this describes the general architecture of a Web crawler and studies how a crawler works. It describes the architecture of the Compaq SRC crawler and its major design goals. Some of these studies briefly describe how the crawling task is parallelized
2.	L. Page and S. Brin.	"The anatomy of a large-scale hypertextual web search engine."	It describes a crawler that distributes individual URLs to multiple machines, which download Web pages in parallel. The downloaded pages are then sent to a central machine, on which links are extracted and sent back to the crawling machines.
3.	J. Cho and H. Garcia-Molina	" Synchronizing a database to improve freshness"	Web crawlers need to update the downloaded pages periodically, in order to maintain the pages up to date. The studies in this discuss various page revisit policies to maximize the "freshness" of the downloaded pages. Studies how a crawler should adjust revisit frequencies for pages when the pages change at different rates.
4.	S. Chakrabarti, M. van den Berg, and B. Dom.	"Focused crawling: A new approach to topic-specific web resource discovery"	Since many crawlers can download only a small subset of the Web, crawlers need to carefully decide what page to download. By retrieving "important" or "relevant" pages early, a crawler may improve the "quality" of the downloaded pages. The study in this category explore how a crawler can discover and identify "important" pages early, and propose various algorithms to achieve this goal.
5.	Md. Abu Kausar and V. S. Dhaka	"An Effective Parallel Web Crawler based on Mobile Agent and Incremental Crawling"	In this paper, a novel incremental parallel Web crawler based on focused crawling is proposed, which can crawl the Web pages that are relevant to multiple pre-defined topics concurrently. Furthermore, to solve the issue of URL distribution, a compound decision model based on multi-objective decision making method is introduced.
6.	Qiuyan HUANG	"Novel Incremental Parallel Web Crawler based on Focused Crawling"	This paper describes an incremental crawler downloads customized contents only from the web for a search engine, thereby helps falling the network load. This network load farther will be reduced by using mobile agents. It is reported in the previous literature that the 40% of the current Internet traffic and bandwidth utilization is due to these crawlers.
7.	Nidhi Grover, Ritika Wason	Comparative Analysis Of Pagerank And HITS Algorithms	In this paper, author has compared two popular web page ranking algorithms namely: HITS algorithm and PageRank algorithm. The paper highlights their variations, respective strengths, weaknesses and carefully analyzes both these algorithms using simulations developed for both.
8.	Trupti V. Udapure, Ravindra D. Kale, Rajesh C. Dharmik	Study of Web Crawler and its Different Types	In this paper author has given The overview of different crawling technologies has been presented in this paper. When only information about a predefined topic set is required, "focused crawling" technology is being used. Compared to other crawling technology the Focused Crawling technology is designed for advanced web users focuses on particular topic and it does not waste resources on irrelevant material.

9.	Pooja Sharma Deepak Tyagi Pawan Bhadana	Weighted Page Content Rank for Ordering Web Search Result	In this paper we focused that PageRank and Weighted PageRank algorithms are used by many search engines but the users may not get the required relevant documents easily on the top few pages. With a view to resolve the problems found in both algorithms, a new algorithm called Weighted Page Content Rank has been proposed which employs Web structure mining as well as Web content mining techniques. This algorithm is aimed at improving the order of the pages in the result list so that the user may get the relevant and important pages easily in the list.

Table 1:

In spite of this enormous amount of effort because of the commercial value of the developed applications, it is still not easy to obtain robust and customizable crawling software. There also exists a significant body of literature studying the general problem of parallel and distributed computing. Some of these studies focus on the design of proficient parallel algorithms.

III. PROBLEM IDENTIFICATION

A search engine is the most popular information retrieval tool. For making efficient IR tool there is need of valuable and swift crawler which having the following objectives:

1) It should download & explore web documents from WWW as much as possible.
2) It should bring high quality documents so that the user gets the required pertinent information within satisfactory time.
3) The documents must be displayed in the order of their relevance with regard to the user query.
4) As the web documents are very much active in nature, search engine should update its repository as frequently as possible. The ideal case would be of synchronizing updation of repository with the web document's tangible change frequency.

To assure the first objective i.e. to cover the Web as much as possible, nowadays search engines do not depend on a single but on multiple crawlers that execute in parallel to achieve the target. While working in parallel, crawlers still face many challenging problems such as overlapping, quality and network bandwidth that need to be addressed.

A. Overlap: Download The Same Page Multiple Times.

1) Need to coordinate between the processes to minimize overlap.
2) To save network bandwidth and increase the crawler's effectiveness.

B. Quality: Download "Important" Pages First.

1) To maximize the "quality" of the downloaded collection.
2) Each process may not be aware of whole web image, and make a poor crawling decision based on its own image of the web.
3) Make sure that the quality of downloaded pages is as good for a parallel crawler as for a centralized.

C. Communication Bandwidth: To Prevent Overlap And Improve Quality.

1) Need to periodically communicate to coordinate.
2) Communication grows significantly as number of crawling processes increases.
3) Need to minimize communication overhead while maintaining effectiveness of crawler.

D. There Are Some Advantage Of Parallel Crawler Which Are As Follows:

Scalability:

1) Due to enormous size of the web, it is imperative to run a parallel crawler.
2) A single-process crawler simply cannot achieve the required download rate.

Search engines employ ranking algorithms to meet second and third objectives mentioned above. Though back link count helps in professionally displaying the documents in the order of their relevance, it fails to bring quality documents.

IV. PROPOSED METHODOLOGY

A sequential (chronological) crawling loop devotes a large amount of time in which either the CPU is idle (during network/disk access) or the network boundary is idle (during CPU processes).

Multi-threading, where each thread tails a crawling loop, can provide sensible speed-up and effectual use of obtainable bandwidth. Figure 4.3 shows our multi-threaded version of the basic crawler. Note that every thread twitches by locking the URL list to pick the next URL to crawl. Afterward selection a URL it unravels the URL list permitting other threads to admittance it. The URL list is again sealed when new URLs are added to it.

Crawling can be explained as a graph as shown in figure 2. The Web is seen as a large graph with pages at its nodes and hyperlinks as its edges. A crawler starts at node (seed) and then traverse the edges to reach other nodes. The process of fetching a page and extracting the links within it is analogous to expanding a node in graph search.

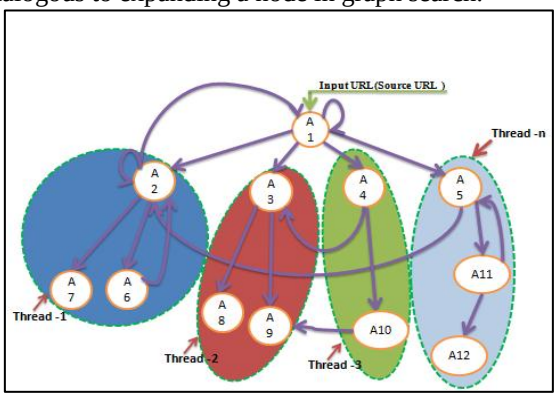

Fig. 2: Crawling as Graph Representation

A. Multithreaded Crawling Algorithm:


```
Step-1.theCrawler.startLink:=new Link(url, level)  /*Set the start link and number of levels to crawl*/
Step-2. theCrawler.allCurrentLevelLinks.add(theCrawler.startLink)/* Add to the list to start traversing */
Step-3. /* Crawl through as many levels as given */
        levels := 0
        while ( levels < theCrawler.levelsToCrawl)
                /* Process each link on the current level using the executor */
Step-4.         while (theCrawler.allCurrentLevelLinks)
Step-5.                 Prank:=PR(theCrawler.currentLink)//Current url PR
Step-6.                 if(prank>InputPR)
Step-7.                         theCrawler.executor.execute(currentLink)
                        End if
                End while //step-4
        End while
Step-8.  theCrawler.executor.shutdown
Step-9.  while (theCrawler.executor.isnotTerminated)
        // wait untill the threads finish
        End while
        /* * Processing has finished. Make note of what links were already traversed, then clear the list of
allCurrentLevelLinks, and add the next level links to it. Start again for the next level. */
Step-10.theCrawler.allTraversedLinks.addAll(theCrawler.allCurrentLevelLinks)
```

V. EXPERIMENTAL RESULTS WITH SCREENSHOTS

In this section we will go through our project first GUI which provide input like seed url, level to crawl and Input PR value for bypass.

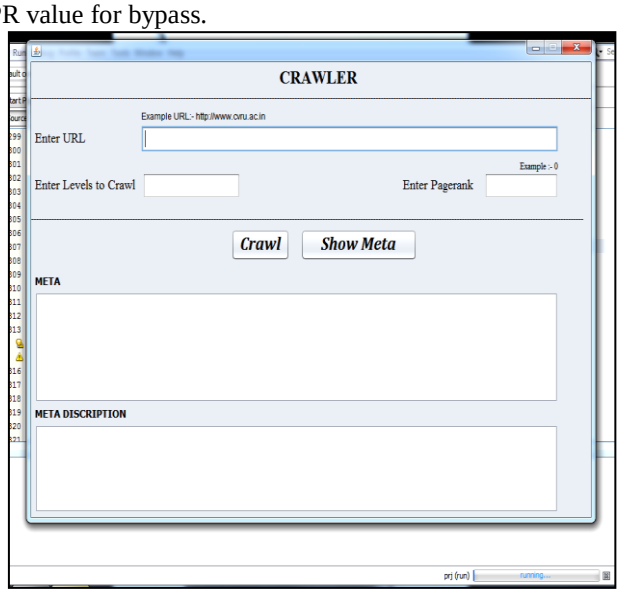

Fig. 3: Main GUI

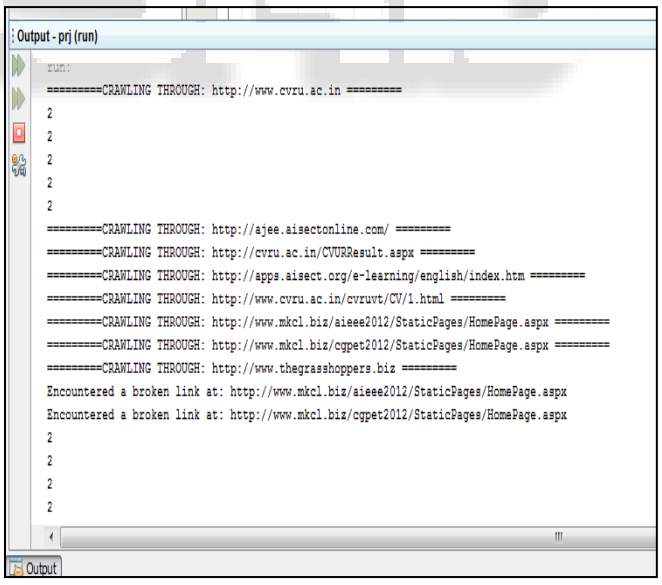

Fig. 4: Output Console during Crawling

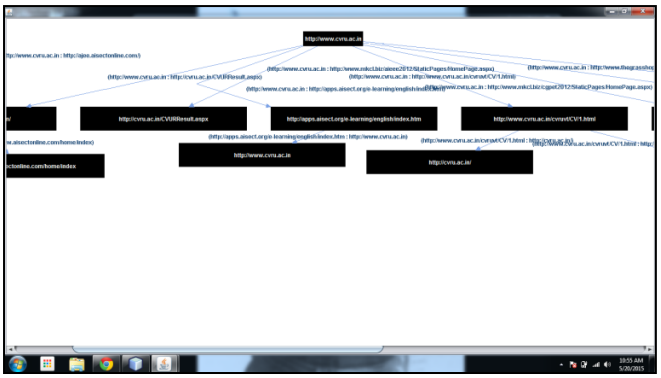

Fig. 5: Output of Crawler as Graph

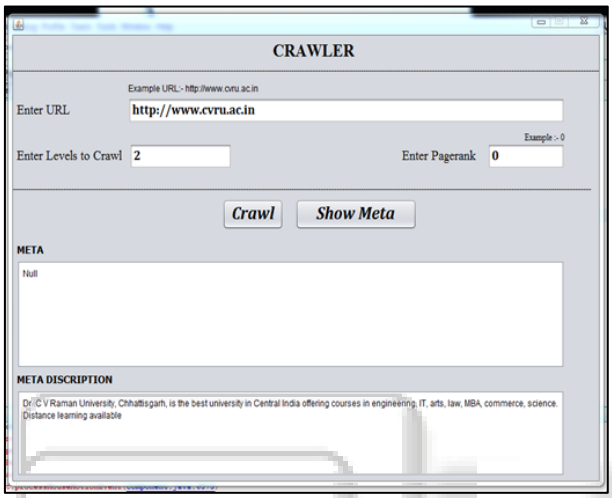

Fig. 6: Shows Meta Description and Keyword

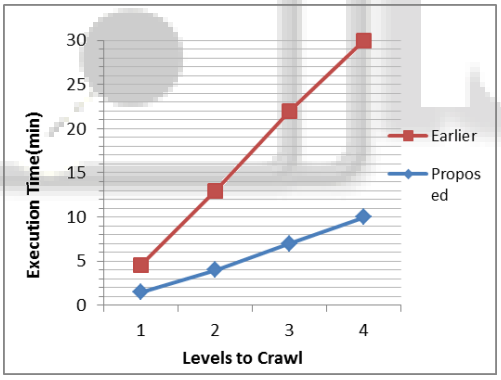

Fig. 7: Performance Comparison of Earlier and Proposed Approach

VI. Conclusion

World Wide Web (WWW) is an enormously powerful resource. It contains a enormous amount of related and unrelated information. Hence, there is a great prerequisite to have algorithms that could list relevant web pages accurately and efficiently on the top of few pages. In spite of the vast amount of both theoretical and practical research on information retrieval, the search problem is still far from being solved. Hnce an effectual crawling mechanism in which integration of multithreaded crawler and Google Numerical weighting technique has been proposed. Numerical weight of webpage is a "vote" by all other pages on the web. By applying PR (Page Rank) it should bring high quality documents so that the user gets the required pertinent information within satisfactory time.

References

[1] David Eichmann, "The RBSE Spider – Balancing effective search against web load", Repository Based Software Engineering Program , Research Institute for Computing and Information Systems, University of Houston – Clear Lake.

[2] Junghoo Cho & Hector Garcia-Molina, "Parallel Crawlers". Proceedings of the 11th international conference on World Wide Web WWW '02, Honolulu, Hawaii, USA. ACM Press. Page(s): 124 – 135.

[3] Spark Jones, K., Walker, S. & Robertson, S. E., 2000. A probabilistic model of information retrieval: Development and comparative experiments (Parts 1 & 2). Information Processing and Management, Vol. 36, Issue 6, pp. 779-808, 809-840.

[4] Haveliwala, T.H. 2002. Topic-sensitive PageRank. Proc. of the 11th International WWW Conference, Honolulu, Hawaii, USA (May 6-11, 2002)

[5] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C.L. & Gori, M. 2000. Focused crawling using context graphs. Proc. of the 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, 2000, pp. 527-534.

[6] Chakrabarti, S. 2003. Mining the Web: Discovering Knowledge from Hypertext Data. San Francisco, CA, USA: Morgan Kaufmann Publishers.

[7] Amento, B., Loren, T. & Hill, W., 2000. Does "authority" mean Quality? Predicting expert quality ratings of Web documents. In Proceedings of SIGIR 2000, Athens, Greece.

[8] Taher H. Haveliwala. Effciient computation of PageRank. Stanford University Technical Report, 1999.

[9] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. Word Wide Web, 2(4):219–229, December 1999.

[10] L. Page and S. Brin. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the Seventh World-Wide Web Conference, 1998.

[11] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In Proceedings of the 2000 ACM SIGMOD, 2000.

[12] Edward G. Coffman, Jr., Zhen Liu, and Richard R. Weber. Optimal robot scheduling for web search engines. Journal of Scheduling, 1(1):15–29, June 1998.

[13] James Gwertzman and Margo Seltzer. World-Wide Web cache consistency. In Proceedings of USENIX 1996 Annual Technical Conference, San Diego, California, January 1996.

[14] Craig E. Wills and Mikhail Mikhailov. Towards a better understanding of web resources and server responses for improved caching. In Proceedings of the Eighth International World-Wide Web Conference, Toronto, Canada, May 1999.

[15] James Pitkow and Peter Pirolli. Life, death, and lawfulness on the electronic frontier. In Proceedings of the Conference on Human Factors in Computing Systems CHI'97, pages 383–390, Atlanta, Georgia, March 1997

[16] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. "Efficient crawling through URL ordering" In

Proceedings of the Seventh International World- Wide Web Conference, Brisbane, Australia, April 1998.
[17] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Building domain-specific search engines with machine learning techniques. In AAAI Spring Symposium on Intelligent Agents in Cyberspace 1999, Stanford, CA, March 1999.
[18] Lawrence Page and Sergey Brin. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the Seventh International World-Wide Web Conference, Brisbane, Australia, April 1998.
[19] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. In Proceedings of the Eighth International World-Wide Web Conference, pages 219–229, Toronto, Canada, May 1999
[20] Md. Abu Kausar and V. S. Dhaka An Effective Parallel Web Crawler based on Mobile Agent and Incremental Crawling in the Journal of Industrial and Intelligent Information Vol. 1, No. 1, March
[21] Qiuyan HUANG Novel Incremental Parallel Web Crawler based on Focused Crawling in Journal of Computational Information Systems 9.
[22] International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 8, October – 2012.

