# Survey on Various Web Crawlers Along With Design Issues

**Shubham Sharma[1], Shivkant[2]**
[1]Dept of Computer Science and Engineering
[2]Assistant Professor, Dept of Computer Science and Engineering
[1, 2] SKITM, Bahadurgarh, India

***Abstract-*** *The number of web pages is increasing into millions and trillions around the world. To make searching much easier for users, web search engines came into existence. Web Search engines are used to find specific information on the World Wide Web. Without search engines, it would be almost impossible for us to locate anything on the Web unless or until we know a specific URL address. Every search engine maintains a central repository or databases of HTML documents in indexed form. Whenever a user query comes, searching is performed within that database of indexed web pages. The size of repository of every search engine can't accommodate each and every page available on the WWW. So it is desired that only the most relevant pages are stored in the database so as to increase the efficiency of search engines. To store most relevant pages from the World Wide Web, a suitable and better approach has to be followed by the search engines. This database of HTML documents is maintained by special software. The software that traverses web for capturing pages is called "Crawlers" or "Spiders".In this paper, we discuss the basics of crawlers and the commonly used techniques and some design issues of crawling the web.*

## I. INTRODUCTION

A web-crawler is a program/software or automated script which browses the World Wide Web in a methodical, automated manner.The structure of the World Wide Web is a graphical structure, *i.e.*, the links given in a page can be used to open other web pages. Actually, Internet is a directed graph, webpage as node and hyperlink as edge, so the search operation could be abstracted as a process of traversing directed graph. By following the linked structure of the Web, we can traverse a number of new web-pages starting from a Starting webpage. Web crawlers are the programs or software that uses the graphical structure of the Web to move from page to page [1]. Such programs are also called wanderers, robots, spiders, and worms. Web crawlers are designed to retrieve Web pages and add them or their representations to local repository/databases. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages that will help in fast searches. Web search engines work by storing information about many web pages, which they retrieve from the WWW itself. These pages are retrieved by a Web crawler (sometimes also known as a spider) — which is an automated Web browser that follows every link it sees. Web *crawlers* are programs that exploit the graph structure of the web to move from page to page. It may be observed that `crawlers' itself doesn't indicate speed of these programs, as they can be considerably fast working programs.Web crawlers are software systems that use the text and links on web pages to create search indexes of the pages, using the HTML links to follow or crawl the connections between pages.

## II. LITERATURE SURVEY

Web crawlers are almost as old as the web itself. The first crawler, Matthew Gray's Wanderer, was written in the spring of 1993, roughly coinciding with the first release of NCSA Mosaic. Several papers about web crawling were presented at the first two World Wide Web conferences. However, at the time, the web was three to four orders of magnitude smaller than it is today, so those systems did not address the scaling problems inherent in a crawl of today's web.

Obviously, all of the popular search engines use crawlers that must scale up to substantial portions of the web. However, due to the competitive nature of the search engine business, the designs of these crawlers have not been publicly described. There are two notable exceptions: the Google crawler and the Internet Archive crawler. The original Google crawler [2] (developed at Stanford) consisted of five functional components running in different processes. A *URL server process* read URLs out of a file and forwarded them to multiple crawler processes. Each *crawler process* ran on a different machine, was single-threaded, and used asynchronous I/O to fetch data from up to 300 web servers in parallel [3]. The crawlers transmitted downloaded pages to a single *StoreServer process*, which compressed the pages and stored them to disk. The pages were then read back from disk by an *indexer process*, which extracted links from HTML pages and saved them to a different disk file. A *URL resolver process* read the link file, derelativized the URLs contained

therein, and saved the absolute URLs to the disk file that was read by the URL server [4]. Typically, three to four crawler machines were used, so the entire system required between four and eight machines [5].

Research on web crawling continues at Stanford even after Google has been transformed into a commercial effort. The Stanford WebBase project has implemented a high performance distributed crawler, capable of downloading 50 to 100 documents per second .Cho and others have also developed models of document update frequencies to inform the download schedule of incremental crawlers.

The Internet Archive also used multiple machines to crawl the web. Each crawler process was assigned up to 64 sites to crawl, and no site was assigned to more than one crawler. Each single-threaded crawler process read a list of seed URLs for its assigned sites from disk into per-site queues, and then used asynchronous I/O to fetch pages from these queues in parallel. Once a page was downloaded, the crawler extracted the links contained in it [6]. If a link referred to the site of the page it was contained in, it was added to the appropriate site queue; otherwise it was logged to disk. Periodically, a batch process merged these logged "cross-site" URLs into the site-specific seed sets, filtering out duplicates in the process [7].

## III. DESIGNING ISSUES OF WEB-CRAWLER

The crawler module retrieves pages from the Web for later analysis by the indexing module. As discussed above, a crawler module typically starts off with an initial set of URLs called, *Seed*. Roughly, it first places *Seed* in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Given the enormous size and the change rate of the Web, many issues arise, including the following:

**What pages should the crawler download?** In most cases, the crawler cannot download all pages on the Web. Even the most comprehensive search engine currently indexes a small fraction of the entire Web. Given this fact, it is important for the crawler to carefully select the pages and to visit "important" pages first by prioritizing the URLs in the queue properly, so that the fraction of the Web that is visited (and kept up-to-date) is more meaningful.

**How should the crawler refresh pages?** Once the crawler as downloaded a significant number of pages, it has to start *revisiting* the downloaded pages in order to detect changes and refresh the downloaded collection. Because Web pages are changing at very different rates, the crawler needs to carefully decide what page to revisit and what page to skip, because this decision may significantly impact the "freshness" of the downloaded collection. For example, if a certain page rarely changes, the crawler may want to revisit the page less often, in order to visit more frequently changing ones [8].

**How should the load on the visited Web sites be minimized?** When the crawler collects pages from the Web, it consumes resources belonging to other organizations. For example, when the crawler downloads page *p* on site *S*, the site needs to retrieve page *p* from its file system, consuming disk and CPU resource. Also, after this retrieval the page needs to be transferred through the network, which is another resource, shared by multiple organizations. The crawler should minimize its impact on these resources. Otherwise, the administrators of the Web site or a particular network may complain and sometimes completely block access by the crawler [10].

**How should the crawling process be parallelized?** Due to the enormous size of the Web, crawlers often run on multiple machines and download pages in parallel. This parallelization is often necessary in order to download a large number of pages in a reasonable amount of time. Clearly these parallel crawlers should be coordinated properly, so that different crawlers do not visit the same Web site multiple times, and the adopted crawling policy should be strictly enforced. The coordination can incur significant communication overhead, limiting the number of simultaneous crawlers [12].

## IV. CONCLUSION & FUTURE WORK

Internet is one of the easiest sources available in present days for searching and accessing any sort of data from the entire world. The structure of the World Wide Web is a graphical structure, and the links given in a page can be used to open other web pages. In this thesis, we have used the graphical structure to process certain traversing algorithms used in the search engines by the Crawlers. Each webpage can be considered as node and hyperlink as edge, so the search operation could be abstracted as a process of traversing directed graph. By following the linked structure of the Web, we can traverse a number of new web-pages starting from a Starting webpage. Web crawlers are the programs or software that uses the graphical structure of the Web to move from page to page. In this thesis, we have briefly discussed about Internet, Search Engines, Crawlers and then Crawling Algorithms.

There are number of crawling strategies used by various search engines. The basic crawling approach uses simple Breadth First method of graph traversing. But there are certain disadvantages of BFS since it is a blind traversing approach. To make traversing more relevant and fast, some heuristic approaches are followed. The results of all the crawling approaches are giving different results.

These heuristic searches keep a check on the relevancy factor of every page to be traversed. Thus the efficiency of the database of the search engine increases and only relevant pages are stored in it.The fast and more accurate version of Fish Search is known as – Shark Search, which is probably the best crawling approach. But there are some issues related with the implementation of Shark Search Approach in simple programming environment as there is lot of factors, calculations associated with it.

In future, work can be done to improve the efficiency of algorithms and accuracy and timeliness of the search engines. The work of Shark Search can be extended further to make Web crawling much faster and more accurate.

## REFERENCES


[1] World Internet Usage and population statistics available at- http://www.internetworldstats.com/stats.htm

[2] Arvind Arasu, Junghoo Cho, Andreas Paepcke: "Searching the Web**",** Computer Science Department, Stanford University.

[3] David Hawking, "Web Search Engines: Part 1", June 2019, available at- http://ieeexplore.ieee.org/iel5/2/34424/01642621.pdf

[4] Web search engine architecture image available at- http://www.ibm.com/developerworks/web/library/wa-lucene2/

[5] Gautam Pant, Padmini Srinivasan, and Filippo Menczer: "Crawling the Web" available at- http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf

[6] Marc Najork, Allan Heydon SRC Research Report 173, "High-Performance Web Crawling", published by COMPAQ systems research center on September 26, 2021

[7] Sergey Brin and Lawrence Page, "The anatomy of a large-scale hyper textual Web search engine", In *Proceedings of the Seventh International World Wide Web Conference*, pages 107–117, April 2018.

[8] Sandeep Sharma and Ravinder Kumar, **"**Web-Crawlers and Recent Crawling Approaches", in International Conference on Challenges and Development on IT - ICCDIT-2008 held in PCTE, Ludhiana (Punjab) on May 30th, 2019

[9] Junghoo Cho, Hector Garcia-Molina: "Parallel Crawlers**",** 7–11 May 2021, Honolulu,Hawaii, USA.

[10] Articles about Web Crawlers available at- http://en.wikipedia.org/wiki/Web_crawler#Examples_of_Web_crawlers

[11] Marc Najork, Janet L. Wiener," Breadth-first search crawling yields high-quality pages", WWW10 proceedings in May 2-5, 2021, Hong Kong.

[12] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Science Department, Stanford University, Stanford, CA Available at- http://www.his.se/upload/51108/google.pdf