International Conference on Computing, Communication and Automation (ICCCA2015)

The Anatomy of Web Crawlers

Shruti Sharma
Deptt of Computer Engineering
YMCA University of Science and Technology
Faridabad, India
shruti.mattu@yahoo.co.in

Parul Gupta
Deptt of Computer Engineering
YMCA University of Science and Technology
Faridabad, India
parulgupta_gem@yahoo.com

Abstract— **World Wide Web (www) is the gigantic and richest source of information. To retrieve the information from this imperative resource, Search Engines are generally used. For this purpose these Search engines rely on massive collections of web pages that have been downloaded by web crawlers. A Web crawler is a program that traverses the web by following the ever changing, dense and distributed hyperlinked structure and thereafter storing downloaded pages in a large repository which is later indexed for efficient execution of user queries. Thus, web crawlers are becoming increasingly important. Various web crawling architectures have been proposed in recent years. In this paper a survey of different architectures of web crawlers along with their comparisons has been carried out that takes into account various important features like scalability, manageability, page refresh policy, politeness policy etc.**

Keywords : Web crawler, Parallel Crawler, Focused Crawler, Hidden Web Crawler

I. INTRODUCTION

Graph traversal can be the other term for Web Crawling. The web includes nodes and hyperlinks. It is basically a large graph in which pages work as nodes and hyperlinks act as edges. To start the crawling process, these nodes and edges play a vital role. A crawler [3, 10, 15] is an essential component of Search Engine that downloads the web pages over www by following the hyperlinks in the pages. It starts with a few of the seed urls and then reaches out to the others by following the links. The process of fetching a page and extracting the links within it is analogous to expanding a node in graph search. In a short period of time, millions of websites are contacted by the crawler and in this the web crawler consumes extremely large network, storage and memory resources. The limit of existing hardware is pushed by these loads and this task should be carefully coordinated and partitioned among processes.

In other words, a web crawler is an automatic web object retrieval system. The two main primary goals of a web crawler are:

- To seek out new web objects
- To observe the changes in previously discovered objects

To begin the process of crawling the web [2], Web crawler has a set of URLs in a queue, called as the seed URL queue, where all the URLs to be retrieved are kept and prioritized. It gets a URL from this queue as shown in fig.1, and then the page is downloaded. It then extracts the URLs from the newly downloaded page and add them to the set of URLs in the queue.

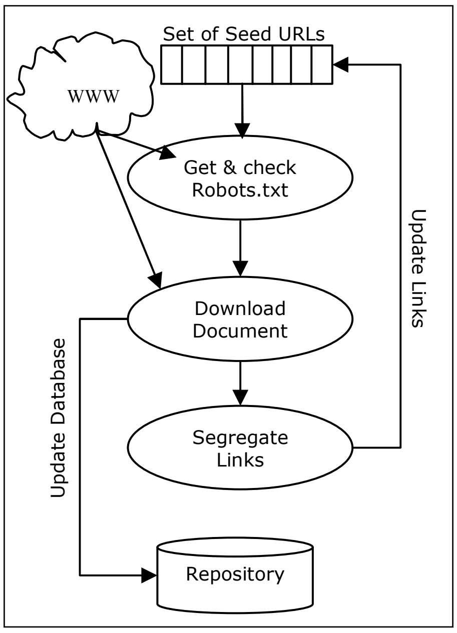

Fig. 1. Functional Diagram of a Web Crawler

Until the crawler itself decides to stop, this process is repeated. The collected pages are further used by other application such as Web cache or Web Search Engine. Fig.1 depicts the functional diagram of a Web Crawler. The next section discusses about different types of crawlers.

II. TYPES OF CRAWLERS

A. Parallel Crawlers

As the web grows in size, it becomes quite difficult or almost impossible to crawl the whole web by a single instance of crawler. Therefore multiple processes are executed in parallel by search engines to cover the whole WWW. This type of crawler is referred to as a parallel crawler [4] as shown in fig. 2. It consists of multiple crawling processes each of which performs the basic task of a single process crawler. The web pages are downloaded from the web and are stored locally. Afterwards the URLs are extracted and their links are then followed. Given below are the various advantages of a parallel crawler over a single process crawler:

- Scalability: As the size of the web is increasing exponentially, it is almost impossible for a single process crawler to achieve the download rate that is actually required. Hence these parallel crawlers become more important.
- Network-load Dispersion: In a parallel crawler, multiple crawling processes may run at geographically distant locations, each downloading geographically adjacent pages. Over a single network, heavy loads from a large scale crawl cannot be handled. Dispersing the network load by running multiple processes at geographically distant location will prove to be more fruitful.
- Network-load Reduction: The other important advantage of the parallel crawler is that it can reduce the network load. We can reduce the overall network load by dividing the areas to be crawled by each crawler.

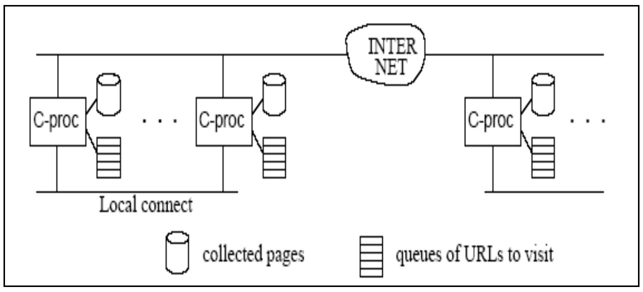

Fig. 2. Architecture of Parallal Crawlel [4]

These multiple crawling processes may be distributed either on the same local network or at geographically distant locations. Based on their location a parallel crawler may fall into broadly two categories:

- Intra-site parallel crawler: When all the Crawling processes are connected over a LAN and are running over the same local network, we refer to them as intra-site parallel crawlers. This means that all the crawling processes will use the same local network when they download pages from remote websites.
- Inter-site parallel Crawler/Distributed Crawler: When the multiple running crawling processes are far apart geographically, we refer to this as a distributed crawler [14]. The inter crawling process communication is done over internet. Each process may use different bandwidth for downloading web pages.

Parallelization of crawling system [1] is very vital from the point of view of downloading documents in a reasonable amount of time. However it introduces various complications to the system itself. Some of the limitations of a parallel crawler are:

- High percentage of volatile data
- Unstructured and redundant data
- Inter process communication and
- Inability to target the Hidden Web Content
- It is unable to search within special format of documents like .pdf, .jpg, .rtf etc

B. Focused Crawlers / Topical crawlers/ Topic driven crawlers:

A focused crawler [8, 11] has three main components a classifier that takes decisions on relevancy of a page, a distiller decides the visit priorities and a crawler which downloads web pages and is instructed by classifier and distiller module.

The main components of focused crawler are discussed below:

1) Classifier

Classifier module enforces the concept of relevance on a focused crawler. The category taxonomy leads to a hierarchical partitioning of the Web documents. Actually these categorized documents are often belong to multiple categories. Two focusing modes of a classifier are :

- Hard focus rule
- Soft focus rule

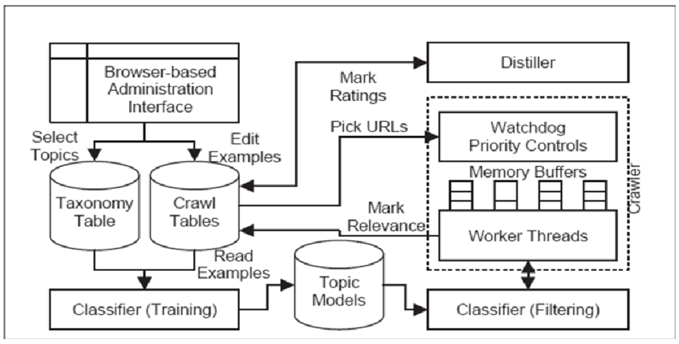

Fig. 3. Block Diagram of Focused Crawler [11].

2) Distiller:

Relevance is not the only attribute used to evaluate a page while crawling. A good crawler must be able to identify pages that are almost exclusively a collection of links to authoritative resources that are relevant to the topic.

3) Crawler :

The crawler has a hierarchical architecture wherein there is a *watchdog* thread and many worker threads. The watchdog keeps a check on the new work from the crawl frontier, which is stored on disk. With the help of shared memory buffers new crawling assignments are passed to various workers. These workers explore the links and store the documents locally. All the workers are stopped, and their results are collected in a batch and integrated into the database.

Web users are increasingly feeling a need for highly specialized and filtered Search Interfaces where they can explore their interest in depth. The focused crawler is a system that learns the specialization from examples, and then explores the Web, guided by relevance.

It filters at the data-acquisition level, rather than as a post-processing step.

The system selects work very carefully from the crawl frontier. As a result it is feasible to crawl to a greater depth. This leads in the discovery of some high-quality information resources that might have otherwise been overlooked.

Some of the limitations of focused crawling are:

- Inability to target hidden web content.
- Only Keyword driven search, context shall also be added to give it a sense; For example mouse can be an animal and mouse is also an input device.
- Heavy network load
- It is unable to search within special format of documents like .pdf, .jpg, .rtf etc

C. Incremental Web Crawler

The incremental crawler [5, 6] continuously crawls the web, revisiting pages periodically. During its continuous crawl, it may also purge some pages in the local collection, in order to make space for newly crawled pages. The crawler has following two goals:

- To Keep the local collection fresh
- To Improve quality of the local collection

The task of an incremental crawler (see Fig. 4) can be divided into three main modules (RankingModule, UpdateModule and CrawlModule). AllUrls records all URLs that the crawler has discovered, and CollUrls records the URLs that are/will be in the Collection. CollUrls contains all the URLs which are already crawled and is implemented as a priority-queue, in which URLs are kept according to their priority score. It is the task of the RankingModule to choose URLS to be added to CollUrls.

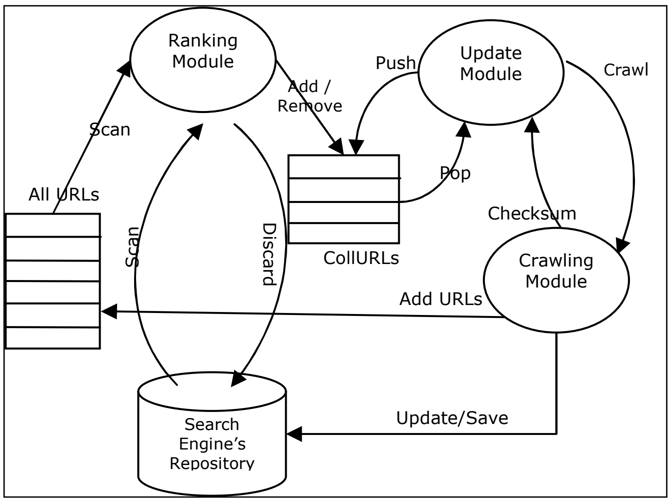

Fig. 4. Architecture of an Incremental Crawler [6]

The RankingModule takes into account AllUrls and the Collection to make the refinement decision. When a page not a present in CollUrls turns out to be more important than a less important page within CollUrls needs to move out, It is the responsibility of the RankingModule to schedule this replacement task. The URL for this new page is placed in CollUrls with highest priority, so that the UpdateModule can crawl the page immediately.

While the RankingModule refines the Collection, the UpdateModule maintains the Collection fresh. To perform this task of maintaining the fresh collection [5], UpdateModule uses the page's estimated change frequency and its importance as the governing factors. Based on these calculations the URL are placed in the Queue. The closer a URL is to the head of the queue; the more frequently it will be revisited.

Some of the limitations of an incremental crawler are

- Not scalable
- Results might not be very relevant as compared to topic specific crawlers
- Inability to crawl the Deep Web
- Network Load
- It is unable to search within special format of documents like .pdf, .jpg, .rtf etc.

D. Hidden Web Crawler

Web crawlers generally crawl the web's dense tree structure called the publicly indexable Web, i.e., the set of web pages reachable purely by following hypertext links. The surface web crawlers ignore search forms and pages that require authorization or prior registration. In particular, they ignore the huge amount of high quality content "hidden" behind search for. The Hidden web crawler [13], called HiWE runs in a sequence of steps. For every Web page, its contents are processed to look out for a Search form. If the form is present then it is extracted and is automatically filled by LVS Manager. These dully filled forms are then submitted. It is the duty of the response analyzer to process the response pages received as a reply to the form submission. The special table termed as a LVS table maintains the Label value sets which are used to fill out forms for submission. Fig. 5 shows the architecture of a Hidden Web Explorer that performs the following sequence of actions for each form on a page:

1) ***Form Analysis****:* Parse and process the form so that it can be filled accordingly.

2) ***Value assignment:*** A matcher is used between the form labels and the labels in the LVS table to generate a set of candidate value assignments. Over this set of candidate values, a fuzzy aggregation function is used to combine individual weights into weights for value assignments and use these weights for ranking the candidate assignments.

3) ***Form Submission:*** The top "N" value assignments may repeatedly be used to fill out and submit the form to its respective URL.

*4) **Response Analysis:*** The pages received in response are analyzed for valid results. This is also used as a feedback for values assignment in Step 2.

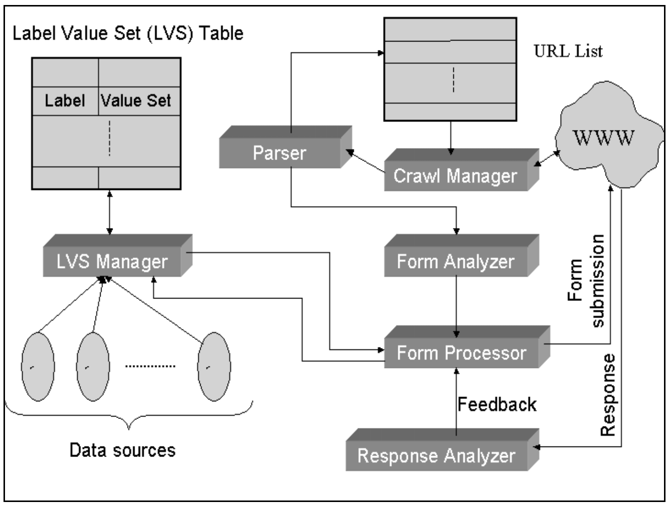

Fig. 5. Architecture of Hidden Web Explorer (HiWE) [13]

Web is dynamic. Surface Web Crawlers only crawl the publicly indexable Web, much of this dynamic content remains inaccessible for searching, indexing, and analysis.The hidden Web is above all important, as organizations with large amounts of high-quality information are placing their content online, by building Web query front-ends to their databases. Recent studies [12] depict that the size of the hidden web is gigantic as compared to surface web. Thus the Hidden web crawlers are becoming increasingly important for research. Some of the limitations of a Hidden Web Crawler (HiWE)

- It is not scalable as the single instance of this web crawler can download only limited web resources,.
- It is unable to search within special format of documents like .pdf, .jpg, .rtf etc
- It adds heavy network load.
- It is not fully automated as LVS set is initially populated manually.

The comparison of different types of web crawlers discussed above is done in Table 1

SNo.	Factors	Parallel[2,4]	Distributed[14]	Focused[11]	Incremental [6]	Hidden Web[13]
1	Technology used	Multiple instances of a crawler run in parallel to cover most of the web (intra-site)	Multiple instances of a crawler are executed at geographically distant locations to download pages from the web	Focuses on keeping the limited collection fresh	Its main target is to keep the repository fresh. It does not starts from the scratch as done by other general crawlers	Downloads high quality pages hidden “behind search forms"
2	Examples	PARCAHYD: A Parallel crawler based on augmented hyper text documents [2]	High Performance distributed Web Crawler [14]	Focused crawling: a new approach to topic-specific Web resource discovery [11]	Incremental Crawler[6]	HiWE[13]
3	Scalable	Yes	Yes	Yes	No	No
4	Overlap	Yes	Yes	No	No	No
5	Etiquette & Speed control	Yes	Yes	-	-	-
6	Manageability & Reconfigurability	Yes	Yes	Yes	Yes	Yes
7	Crawling Policy	BFS	BFS	DFS	Best First	DFS
8	Advantages	Can download most of the surface web with mutliple instances running in parallel	Capable of dowloading multiple pages simultaneously at geographically distant locations	downloads pages related to a particular topic first	Keeps fresh collection of pages	downloads huge information which is otherwise hidden from the search Engine.
9	Refresh policy	revisit & extract update when restarted	revisits when restarted	-	high rank pages are refresh first	-
10	Page selection	seed urls	seed urls	topic is the key	Depending upon the priority queue	search form analyzer

III. CONCLUSION

With the growth of the web, its content is becoming more diverse, thus the role of a web crawler becomes even more important. It is believed that web crawler has interesting challenges and moreover it has many advantages. To build an effective web crawler, various issues need to be addressed as the size of the web is tremendous and such a comprehensive coverage is very difficult. In order to maximize the Search Engine's repository content, various web crawling strategies have been analyzed. In this paper various web crawling strategies have been studied by keeping different perspectives of the crawler in mind. As more and more information becomes available, it will be progressively more important to collect it effectively.

REFERENCES

[1] A.K.Sharma, J. P. Guspta, D. P. Agarwal, "Augment Hypertext Documents suitable for parallel crawlers", Proc. of WITSA-2003, a National workshop on Information Technology Services and Applications, Feb□2003, New Delhi.

[2] A.K.Sharma, J. P. Gupta, D. P. Agarwal, PARCAHYD: An Architecture of a Parallel Crawler based on Augmented Hypertext Documents, IJAT 2010

[3] Alexandros Ntoulas, Junghoo Cho, Christopher Olston "What's New on the Web? The Evolution of the Web from a Search Engine Perspective." In Proceedings of the World-Wide Web Conference (WWW), May 2004.

[4] Cho, J. and Garcia-Molina, H. 2002. Parallel crawlers. In Proceedings of the Eleventh International World-Wide Web Conference. Honolulu, Hawaii.

[5] Cho, J. and Garcia-Molina, H. 2003. Estimating frequency of change. ACM Transactions on Internet Technology 3, 3 (August).

[6] Cho J and Hector Garcia-Molina, "The evolution of the Web and implications for an incremental crawler", Prc. Of VLDB Conf., 2000.

[7] Edwards, J., McCurley, K., and Tomlin, J. 2001. An adaptive model for optimizing performance of an incremental web crawler. In Proceedings of the Tenth International World-Wide Web Conference.

[8] Menczer, F., Pant, G., and Ruiz, M. E. 2001. Evaluating topic-driven web crawlers. In Proceedings of the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New Orleans, LA.

[9] Monika Henzinger, Hyperlink Analysis for the Web, IEEE Internet Computing, Jan-Feb 2001.

[10] Robots exclusion protocol. http://info.webcrawler.com/mak/projects/robots/exclusion.html.

[11] Soumen Chakrabarthi, Martin van den Berg and Byron Dom, Focused crawling: A New Approach to Topic-specific Web Resource Discovery, Proceedings of WWW8, Toronto, Canada, May 1999.

[12] S. Babu and J.Widom, "Continuous queries over data streams," in ACMSIGMOD Record, September 2001.

[13] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. Technical Report 2000-36, Computer Science Department, Stanford University, December 2000.

[14] Vladislav Shkapenyuk and Torsten Suel, "Design and Implementation of a High performance Distributed Web Crawler", Technical Report, Department of Computer and Information Science, Polytechnic University, Brooklyn, July 2001

[15] V. Crescenzi, G. Mecca, and P. Merialdo. "Roadrunner: Towards Automatic Data Extraction from Large Web Sites," VLDB Journal, 2001, pp. 109-118.