WEB CRAWLERS OF SEARCH ENGINE

Mahender Kumar Beniwal

Triloki Nath Sharma

Arpita Sharma

ABSTRACT-

As the web is expanding abruptly, so need to extract knowledge from the web is becoming essential and is accepted. This is possible because of fortune and ease of information. In order to get web pages search engines are used that are based on Web crawling framework. This paper gives the outline of how the Web crawlers are connected with search engine and also explains the fundamental task performed by the search engine.

Keywords— Web Crawlers, Focused Crawling, Distributed Crawling

INTRODUCTION

The World Wide Web is the universe of network accessible information. WWW on the Web is a service that resides on computers which is connected to the Internet and allows end users to access data that is stored on the computers using standard interface software. Search engine is a computer program that searches for particular keywords and returns a list of documents in which they were found, especially a commercial service that scans documents on the Internet. A search engine finds information for its database by accepting listings sent in by those who want exposure, or by getting the information from their "Web crawlers," "spiders," or "robots," programs that roam the Internet storing links to and information about each page they visit. Web Crawler is a program, which fetches information from the World Wide Web in an automated manner. Web crawling is an important research issue. Crawlers are software components which visit portions of Web trees, according to certain strategies and then collect retrieved objects in local repositories.

A web crawler is a program which when provided with one or more seed URLs, downloads the web pages associated with these URLs, extracts any hyperlinks contained in them, and recursively continues to download the web pages identified by these hyperlinks. Web crawlers are an important component of web search engines, where they are used to collect the mass of web pages indexed by the search engine. They are used in many other applications that process large numbers of web pages, like web data mining, comparison shopping engines. In order to crawl a significant fraction of the "surface web" in a reasonable amount of time, web crawlers must download thousands of pages per second, and are typically distributed over tens or hundreds of computers. Their two main data structures are the "frontier" set of yet-to-be-crawled URLs and the set of discovered URLs which typically do not fit into main memory, so efficient disk-based representations need to be used. Finally, the need is to not to overload any particular web server, and a desire to prioritize the crawl towards high-quality pages.

FUNDAMENTALS

The algorithm executed by a web crawler is very simple: From a set of candidates select a URL, then download the associated web pages, and extract the URLs i.e. hyperlinks contained therein, and add those URLs that have not been encountered before to the candidate set. Certainly, it is somewhat possible to implement a simple functioning web crawler in a few lines of a high-level scripting language such as Perl.

On the other hand, building a web-scale web crawler imposes major engineering challenges, all of which are ultimately related to scale. In order to maintain a search engine body of say, ten billion web pages, in a reasonable state of freshness, say with pages being refreshed every 4 weeks on average, the crawler must download over 4,000 pages/second. In order to achieve this, the crawler must be distributed over multiple computers, and each crawling machine must pursue multiple downloads in parallel. But if a distributed and highly parallel web crawler were to issue many concurrent requests to a single web server, it would in all likelihood overload and crash that web server. Therefore, web crawlers need to implement politeness policies that rate-limit the amount of traffic directed to any particular web server. There are many possible politeness policies; one is to disallow concurrent requests to the same web server which is very simple to implement and another a slightly more sophisticated policy is to wait for time proportional to the last download time before contacting a given web server again.

In some web crawler designs, the page downloading processes are distributed, while the major data structures – the set of discovered URLs and the set of URLs that have to be downloaded – are maintained by a single machine. This design is conceptually simple, but it does not scale indefinitely, eventually the central data structures become a bottleneck. The alternative is to partition the major data structures over the crawling machines. Ideally, this should be done in such a way as to minimize communication between the crawlers. One way to achieve this is to assign URLs to crawling machines based on their host name. Partitioning URLs by host name means that the crawl scheduling decisions entailed by the politeness policies can be made locally, without any communication with peer nodes. Moreover, since most hyperlinks refer to pages on the same web server, the majority of links extracted from downloaded web pages is tested against and added to local data structures, not communicated to peer crawlers.

Once a hyperlink has been extracted from a web page, the crawler needs to test whether this URL has been encountered before, in order to avoid adding multiple instances of the same URL to its set of pending URLs. This requires a data structure that supports set membership test, such as a hash table. Care should be taken that the hash function used is collision-resistant, and that the hash values are large enough. If RAM is not an issue, the table can be maintained in memory otherwise a disk-based implementation must be used. Implementing fast disk-based set membership tests is extremely hard, due to the physical limitations of hard drives. If the URL space is partitioned according to host names among the web crawlers, the set data structure is partitioned in the same way, with each web crawling machine maintaining only the portion of the set containing its hosts. Consequently, an extracted URL that is not maintained by the crawler that extracted it must be sent to the peer crawler responsible for it.

Once it has been determined that a URL has not been previously discovered, it is added to the frontier set containing the URLs that have yet to be downloaded. The frontier set is generally too large to be maintained in main memory. The frontier could be implemented by a simple disk-based FIFO queue, but such a design would make it hard to enforce the politeness policies, and also to prioritize certain URLs over other URLs. URL prioritization could be achieved by using a priority queue implemented as a heap data structure, but a disk-based heap would be far too expensive, since adding and removing a URL would require multiple seek operations. If the URL space is partitioned according to host names among the web crawlers, the frontier data structure is partitioned along the same lines.

CRAWLING TECHNIQUES

A. **Human Powered Directories**

These are built by human selection i.e. they depend on humans to create listings. They are organized into subject categories and subjects do classification of pages. Human powered directories never contain full text of the Web page they link to. They are smaller than most search engines.

B. **Hybrid Search Engine**

A hybrid search engine differs from traditional text oriented search engine such as Google or a directory-based search engine such as Yahoo in which each program operates by comparing a set of metadata, the primary corpus being the metadata derived from a Web crawler or taxonomic analysis of all internet text, and a user search query. In contrast, hybrid search engine may use these two bodies of metadata in addition to one or more sets of metadata that can, for example, include situational metadata derived from the client's network that would model the context awareness of the client.

C. **Crawler Based Search Engines**

Crawler based search engines create their listings automatically. Computer programs 'spiders' build them not by human selection. They are not organized by subject categories; a computer algorithm ranks all pages. Such kinds of search engines are huge and often retrieve a lot of information -- for complex searches it allows to search within the results of a previous search and enables you to refine search results. These types of search engines contain full text of the Web pages they link to. So one can find pages by matching words in the pages one wants.

WORKING OF A WEB CRAWLER

Web crawlers are an essential component to search engines; running a Web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of Web servers and various name servers, which are all beyond the control of the system. Web crawling speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel. Despite the numerous applications for Web crawlers, at the core they are all fundamentally the same. Following is the process by which Web crawlers work:

1. Download the Web page.
2. Parse through the downloaded page and retrieve all the links.
3. For each link retrieved, repeat the process.

The Web crawler can be used for crawling through a whole site on the Inter-/Intranet. You specify a start-URL and the Crawler follows all links found in that HTML page. This usually leads to more links, which will be followed again, and so on. A site can be seen as a tree-structure, the root is the start-URL; all links in that root- HTML-page are direct sons of the root. Subsequent links are then sons of the previous sons. A single URL Server serves lists of URLs to a number of crawlers. Web crawler starts by parsing a specified Web page, noting any hypertext links on that page that point to other Web pages. They then parse those pages for new links, and so on, recursively. WebCrawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. Each crawler keeps roughly 300 connections open at once. This is necessary to retrieve Web pages at a fast enough pace. A crawler resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a Web browser does when the user clicks on links. All the crawler really does is to automate the process of following links. Web crawling can be regarded as processing items in a queue. When the crawler visits a Web page, it extracts links to other Web pages. So the crawler puts these URLs at the end of a queue, and continues crawling to a URL that it removes from the front of the queue.

A. Robot Protocol

The robot.txt file gives directives for excluding a portion of a Web site to be crawled. Analogously, a simple text file can furnish information about the freshness and popularity of published objects. This information permits a crawler to optimize its strategy for refreshing collected data as well as replacing object policy.

B. Resource Constraints

Crawlers consume resources: network bandwidth to download pages, memory to maintain private data structures in support of their algorithms, CPU to evaluate and select URLs, and disk storage to store the text and links of fetched pages as well as other persistent data.

C. Meta Search Engine

A meta-search engine is the kind of search engine that does not have its own database of Web pages. It sends search terms to the databases maintained by other search engines and gives users the results that come from all the search engines queried. Fewer meta searchers allow you to delve into the largest, most useful search engine databases. They tend to return results from smaller and/or free search engines and miscellaneous free directories, often small and highly commercial.

KEY APPLICATIONS

Web crawlers are usually used in web search engines, to collect the pages that are to be indexed. Other than the common search Web Crawlers have many applications, for example in web data mining.

CONCLUSION

Here, in this paper we can conclude that due to the vast size of the whole WWW and to resource availability the complete web crawling coverage cannot be achieved. In order to limit the crawling process over a selected website a kind of threshold is set up which is the number of visited URLs, level in the website tree, compliance with a topic, etc. Therefore to improve the quality of retrieved contents and reducing stale content and missing pages, this information is available in search engines to store or refresh most relevant and updated web pages.

FUTURE DIRECTIONS

Commercial search engines are global companies serving a global audience, and as such they maintain data centers around the world. In order to collect the corpora for these geographically distributed data centers, one could crawl the entire web from one data center and then replicate the crawled pages to the other data centers, one could perform independent crawls at each data center and thus serve different indices to different geographies, or one could perform a single geographically-distributed crawl, where crawlers in a given data center crawl web servers that are (topologically) close-by, and then propagate the crawled pages to their peer data centers. The third solution is the most elegant one, but it has not been explored in the research literature, and it is not clear if existing designs for distributed crawlers would scale to a geographically distributed setting.

REFERENCES

1. Brin S. and Page L. The anatomy of a large-scale hypertextual search engine. In Proc. 7th Int. World Wide Web Conference, 1998, pp. 107–117.
2. Burner M. Crawling towards eternity: building an archive of the World Wide Web. Web Tech. Mag., 2(5):37–40, 1997.
3. J.Cho, H.Garcia-Molina, L.Page, "Efficient Crawling Through URL ordering" , www7/Computer Networks 30 (1-7):161-172(1998)
4. Pant, Gautam, Padmini Srinivasan and Filippo Menczer: Crawling the Web, 2003.
5. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S.Raghavan, "Searching the Web", ACM Transactions on Internet Technology, Vol. 1, Num. 1, August 2001, pp.2-43.
6. S. Chakrabarti, K. Punera, M. Subramanyam, "Accelerated focused crawling through online relevance feedback", WWW 2002, pp. 148-159.
7. Grossan, B. "Search Engines: What they are, how they work, and practical suggestions for getting the most out of them," February1997.
8. C. Aggarwal, F. Al-Garawi, P. Yu, "Intelligent crawling on the World Wide Web with arbitrary predicates", WWW 2001, pp. 96-105.
9. Brian Pinkerton. Finding What People Want: Experiences with the WebCrawler. In Proceedings of the Second International World Wide Web Conference, 1994.