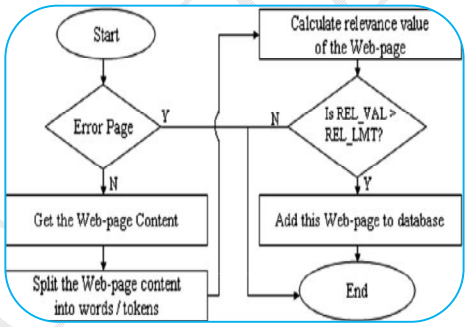
REVIEW OF RESEARCH
ISSN: 2249-894X
IMPACT FACTOR : 5.7631(UIF)
UGC APPROVED JOURNAL NO. 48514
VOLUME - 8 | ISSUE - 10 | JULY - 2019

DOMAIN SPECIFIC PARALLEL CRAWLER

Prof. Santosh Kulkarni
Assistant Professor,
Prin. K.P. Mangalvedhekar Institute of Management,
Career Development & Research , Solapur.

ABSTRACT:

The World Wide Web is a collection of millions of HTML-formatted documents that are linked together. It has become necessary to parallelize a crawling process because it has become difficult to traverse all URLs in web documents and handle these URLs due to the web's growing and dynamic nature. The ecology of crawler workers, which simultaneously download web content, further parallelizes the crawler process. A novel parallel crawler architecture based on domain-specific crawling is proposed in this paper to make the task of crawling more efficient, scalable, and load-sharing among the various crawlers that parallel download web pages related to various domain-specific URLs.



KEYWORDS: *WWW, URLs, crawling process, parallel crawlers.*

INTRODUCTION

A program that retrieves and stores web pages from the Internet is known as a web crawler. A web crawler (M. Burnner et al. al., 1997, pp. 37-40) begins by putting a seed queue of a few URLs. URL (Uniform Resource Locator) is a URI (Uniform Resource Identifier) that specifies the location and method of retrieval of an identified resource, A distributed collaborative hypermedia information system can use the speed and lightness of HTTP. The web crawler downloads a web page, extracts any URLs from the downloaded page, adds the new URLs to the seed queue, and retrieves the following URL from the seed queue. This crawling procedure is repeated until the web crawler decides to stop.

Crawling the entire Web with a single process becomes increasingly challenging or impossible as the Web grows in size. Parallel crawlers, which many search engines use to run multiple processes simultaneously (Cho, J. et al., 2002.).

Although many existing search engines use parallel crawlers internally, very little scientific research has been done on parallel crawler parallelization (Yadav, D. et al., 2007, Yu, C. et. al. According to the available literature (Balamurugan, et. al., 2008, pp. 455-466) al., 2008, Dim, M., 1996) it has been seen that as

1) As the quantity of creeping processes expands, the nature of downloaded pages turns out to be more awful, except if they trade back connect messages frequently.
2) When the crawler downloads only a small portion of the pages, the quality of the firewall mode is significantly lower than that of the single-process crawler.
3) As the number of URL exchanges increases, the communication overhead does not increase in a straight line.

The following structure is used to organize the paper: The related work to web crawling is discussed

in the second section. Section 3 outlines the parallel crawling architecture that has been proposed. Every one of the parts of our design are likewise talked about in a similar segment. Model talked about in area 4. Sections 5 and 6 provide the drawn conclusion and the work to be done in the future, respectively.

Related Work

Web crawler is the significant part of the web. Numerous web crawlers have been developed since their introduction. Wanderer was the first web crawler (Gray, M., 1996). Multiple machines are used by the Internet Archive (Burner, M., 1997, pp. 37-40) crawler to search the web for duplicate URLs. The first version of Google's crawler (Brin, S., et al. al. 1998, P. 107-117) is a distributed system that crawls the web using four to eight machines. WebBase (Cho, J.,et. al., 2006, P. 153-186) a test web vault, has executed a steady crawler (Junghoo Cho, et. al., 2000). Mercator (Heydon, A., et. al., 1999. P. 219-229) lists a number of necessary functional components for the crawler's fundamental algorithm: storing the list of URLs to download, converting host names into IP addresses, utilizing the HTTP protocol to download documents, extracting links from HTML documents, and determining whether a URL has previously been encountered.

The difficult issues of overlap, quality, communication bandwidth, scalability, and network load dispersion and reduction (Cho, J., et al.) al. 2002) hinder the parallel crawler's operation. Due to the aforementioned issues, it is necessary to divide the crawling of web pages among the various parallel crawlers based on specific criteria; to spread the workload among the various processors. The efficiency, access speed, dependability, and concurrency control of such a distribution will rise. A parallel web crawler runs multiple crawling processes simultaneously to maximize the download rate. There are two types of parallel web crawlers: "intrasite" and "distributed." All of the crawling processes are run on the same local network in an "intra-site" crawler.

All of the crawling processes are carried out at geographically distant locations by a "distributed" crawler (Yadav, D, et al. al. 2008, P. 929–940) are connected via WAN or the Internet. A parallel crawler's crawling processes must download web pages from specific domains. Each crawling process is responsible for a predetermined portion of the Web. URLs are divided into internal and external URLs by a web crawler. The downloaded page's internal URLs represent web pages that are part of the domain. A URL that is not an internal URL is called an external URL. According to Joo Yong Lee et al. al. 2008) to handle URLs from other places. in exchange, cross-over, and firewall modes, respectively.

Proposed Architecture

On the basis of intelligent domain-specific crawling, a novel parallel crawler architecture is being proposed. Slithering on this base makes the undertaking more viable as far as significance and burden sharing . the proposed engineering of space explicit clever equal crawler. The search engine interface is where users interact in search of specific information by entering a "keyword." The seed URL is first searched in the Web Cache, which contains the most recently visited URLs. Then, the seed URL is searched in the mother server's repository, which maintains the keyword database and the corresponding URLs as a node-based tree with each node representing a domain. Through the URL distributor, the URL dispatcher sends the seed URL to the relevant DNS Queues, and then the crawling process continues. On the other hand, the search/insert method is used to add the keyword to the database and its corresponding URL is added whenever it is encountered in the future if the search for the URL for the particular key word fails (i.e., the keyword has not yet been visited).

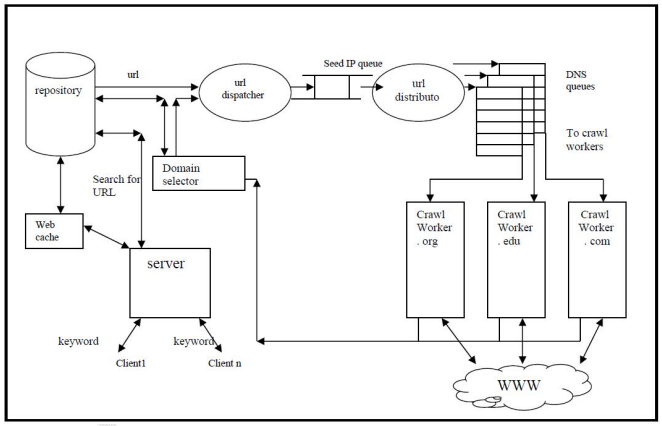
There are a number of domain-specific queues in the architecture for various domains like.edu,.org,.ac, and.com, among others. The respective Crawl Workers receive the URLs for these queues. As a result, parallel crawlers split the load based on specific domains.

Description of the various modules

Crawl Worker

The proposed architecture's most important module is this one. The following steps are required for the crawl module to function:

The crawl worker's various modules are depicted in
(1) Fetch URL;
(2) Download Web page;
(3) Extract URLs;
(4) Analyze URL; and
(5) Forward URL. The URL collector transmits the seed URL to the downloader so that the web pages can be downloaded. The downloaded pages are stored on disk by the URL collector; the removed URLs are put away Top to bottom URL Cradle briefly. If the Depth URL Buffer is full, all of the URLs in it are written to the Depth URL File, which clears the buffer and sends it to the link extractor. The URL analyzer checks for syntax errors and duplicate URLs. The extracted URLs are simultaneously submitted to the domain selector, whose job it is to identify the domain of the URL and store the information in the repository of the mother server. On the other hand, the extracted URLs are simultaneously submitted to the URL Dispatcher for further crawling. The link extractor extracts any URLs in the downloaded pages that are temporarily stored in depth URL buffer.



Proposed architecture of Domain Specific Parallel Crawler

Domain Selector

The domain selector's job is to find the domains of the links that were extracted by the link extractor and forward the URL to the repository for storage in the table for that domain. In contrast, the domain selector also transmits the received URL to the URL dispatcher for further crawling.

The entire HTML of each web page can be found in the repository. The choice of compression method is a compromise between spread and compression ratio, and z lib/ is used to compress each page. Documents are stored sequentially in the repository, preceded by the doc ID, length, and URL. To access the repository, no additional data stretches are required. Development is made much simpler and data consistency is improved as a result. The vault stores the data set for the watchword and the relating URL's as a tree of hubs where every hub is a area. In point of fact, the structure is hierarchical, with the domains and subdomains represented by the parent and child, respectively, as depicted in Figure 3.3. It should be noted that each node has a domain name and the URLs that go with it. New URLs are added to the database so that it can be used in the future. The client's keyword is searched in the database, which has a hierarchical structure and uses the appropriate searching strategy (breadth first) (Najork, M., et al.). al., 2001) is utilized to obtain the appropriate seed URL. When there is a positive response, the right URLs are sent to the URL Dispatcher.

Web Cache

In preparation for future requests, a web cache stores web resources. Web caching works because the most popular resource is more likely to receive future requests. The upsides of Web Reserve are

1) lessens network data transfer capacity use, which can set aside cash for both shopper and the maker.
2) reduces the delay that users perceive, which in turn raises their value perception.
3) reduces the load on the origin servers, resulting in lower hardware and support costs for content providers and a quicker response time for users with no cached resources.

Example

Step1: Through the interface of the search engine, the user enters the keyword. The keyword that was entered into the given example is "Computer Network." Below are snapshots of the given example.

Step2: Figure shows how to search for keywords. 4.2, which leads to links from various Domains like.com,.org, and so on. ,which are appropriated by URL Wholesaler to particular DNS lines ,which are additionally sent to various creep laborers.

Future Work

However, the most important part of the future work will be carrying out the proposed work to improve the parallel crawler's quality. A few fascinating issues have been recognized which should be tended to by future research before the proposed engineering can be carried out. Detailed Queries: In addition to the link extraction queries, we will investigate ways to support more complex crawl queries. Illustration of such questions incorporates disseminated page rank calculation, figuring site rundowns and developing reversed files of the page. Tolerance for Faults:

There are no mechanisms that we have provided to guarantee the crawl's durability in the event of node failure. Others:

The efficient selection of domains for the link extracted from various sites, as well as the rapid distribution of URLs to specific crawlers, improve the overall crawling speed, both in terms of information retrieval and crawling security.

CONCLUSION

For building a parallel crawler on the lines of domain-specific crawling, the following novel architecture is proposed:

(1) Full Distribution: For improved performance, the Domain Specific Crawler (DSCrawler) is distributed among multiple crawling machines, one for each domain. Crawling machines independently download web pages without communicating with one another.

(2) Capacity: DSCrawler's performance can be scaled by adding more machines based on the increase in domains because of its fully distributed architecture, allowing it to handle the rapidly expanding Web.

(3) Burden Adjusting: The URL Distributor distributes the URLs to be crawled to specific Domain Specific Queues, distributing the crawling to various crawlers and balancing the crawling load.
(4) Dependability Because the function of the remaining crawl workers is unaffected by the failure of a single worker, the system as a whole is more reliable when it has multiple crawlers working independently.

REFERENCES

1. Cho, J., Garcia-Molina, H., Haveliwala, T., Lam, W.,Paepcke, A., Raghavan, S., Wesley, G., 2006. Stanford WebBase Components and Applications. In ACM Transactions on Internet Technology. Vol. 6, No. 2,153-186.
2. Gray, M., 1996. Internet Statistics: Growth and Usage ofthe Web and the Internet, http://www.mit.edu/people/mkgray/net/.
3. Heydon, A., Najork, M., 1999. Mercator: A scalable,extensible Web crawler. In World Wide Web, Vol. 2,No. 4, 219-229.
4. Joo Yong Lee, Sang Ho lee," scrawler: a seed-by-seed parallel web crawler", school of computing, soongsil university, seoul, korea,2008.
5. Junghoo Cho and Hector Garcia – Molina, 2000 "The Evolution of the Web and implementation for an incremental crawler" , Prc. Of VLDB Conf..