A Novel Architecture for Domain Specific Parallel Crawler

Nidhi Tyagi & Deepti Gupta
Department of Computer Engineering, Shobhit University, Meerut, India
Email: mnidhity@rediffmail.com & deepti.mrt@gmail.com

Abstract

The World Wide Web is an interlinked collection of billions of documents formatted using HTML. Due to the growing and dynamic nature of the web, it has become a challenge to traverse all URLs in the web documents and handle these URLs, so it has become imperative to parallelize a crawling process. The crawler process is further being parallelized in the form ecology of crawler workers that parallely download information from the web. This paper proposes a novel architecture of parallel crawler, which is based on domain specific crawling, makes crawling task more effective, scalable and load-sharing among the different crawlers which parallel download web pages related to different domains specific URLs.

Keywords: WWW, URLs, crawling process, parallel crawlers,

1. Introduction

A web crawler is a program that retrieves and stores web pages from the Web. A web crawler (Burner, M. et. al., 1997, P. 37-40) starts by placing an initial set of URLs in a seed queue. URL (Uniform Resource Locator) is a URI (Uniform Resource Identifier) that specifies where an identified resource is available and the mechanism for retrieving it, for example http://www.w3.org/default.aspx. HTTP is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. 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. As the size of the Web grows, it becomes more difficult or impossible to crawl the entire Web by a single process. Many search engines run multiple processes in parallel which are referred as parallel crawler (Cho, J. et.al., 2002.). While many existing search engines already use a parallel crawler internally, there has been little scientific research conducted on the parallelization of a crawler except few (Yadav, D. et.al., 2007, Yu, C. et. al.,2008, P. 455-466) As from the available literature (Balamurugan, et. al., 2008, Gray, M., 1996) it has been found that
1) As the number of crawling processes increases, the quality of downloaded pages becomes worse, unless they exchange back link messages often.
2) The quality of the firewall mode is significantly worse than that of the single-process crawler when the crawler downloads a relatively small fraction of the pages.
3) The communication overhead does not increase linearly as the number of URL exchange increases.

The paper is organized in the following way. In next section 2, the related work to the web crawling is discussed. The proposed architecture for parallel crawling is given in section 3. All the components of our architecture are also discussed in the same section. Example discussed in section 4. Conclusion drawn and the future work are given in section 5 and 6 respectively.

2. Related work

Web crawler is the major component of the web. Since the invention of the web many crawlers have been introduced. The first web crawler is Wanderer (Gray, M., 1996). The Internet Archive (Burner, M., 1997,P. 37-40) crawler uses multiple machines to crawl the web and find duplicate URLs. The original Google crawler (Brin, S., et. al. 1998, P. 107- 117) is a distributed system that uses four to eight machines for web crawling. WebBase (Cho, J.,et. al., 2006, P. 153-186) an experimental web repository, has implemented an incremental crawler (Junghoo Cho, et. al., 2000). Mercator (Heydon, A., et. al., 1999. P. 219-229) presents a number of functional components that crawler's basic algorithm requires: storing the list of URLs to download, resolving host names into IP addresses, downloading documents using the HTTP protocol, extracting links from HTML documents, and determining whether a URL has been encountered before.

The challenging issues like overlap, quality, communication bandwidth, scalability, Network-load dispersion and Network-load reduction (Cho, J., et. al. 2002) make the working of the parallel crawler difficult. Considering the above issues, there is a need to distribute the task of crawling the web pages among the different parallel crawlers on the specific criteria; to distribute the load of the different processors. Such a distribution will

improve the efficiency, accessing speed, reliability and concurrency control. In order to maximize the download rate, a parallel web crawler runs multiple crawling processes simultaneously. A parallel web crawler can be "intra-site" or "distributed ". An "intra –site" crawler runs all crawling processes on the same local network such as LAN. A "distributed" crawler runs all crawling processes at geographically distant locations (Yadav, D, et. al. 2008, P. 929-940) connected by the Internet or WAN. Each crawling process of a parallel crawler is responsible for a pre-determined partition of the Web .Each crawling process has to download web pages on specified domains. A web crawler separates URLs into internal URLs and external URLs. The internal URLs represent web pages that belong to the domain, to which the downloaded page belongs. An external URL is a URL that is not an internal URL. There are three modes (Joo Yong Lee et. al. 2008) to handle external URLs. In the "firewall mode", "cross-over mode" and "exchange" mode.

The proposed novel architecture for building a parallel crawler is based on the domain specific crawling (DS Parallel Crawler). DS Parallel crawler handles external URLs in the "firewall mode" and overcome the limitations of firewall mode by distributing the external URLs to domain specific crawler which are identified by the domain selector.

3. Proposed Architecture

A Novel architecture of parallel crawler which based on intelligent domain specific crawling is being proposed. Crawling on this base makes the task more effective in terms of relevancy and load sharing .The fig 3.1, shows the proposed architecture of domain specific intelligent parallel crawler. The users interact through the search engine interface, for the specific information in the form of 'keyword'. The seed URL is first searched in the Web Cache, which contains the recently visited URLs, else the search is made in the repository of the mother server, which maintains the database for the keyword and the corresponding URL's as a tree of nodes where each node is a domain. The URL dispatcher, dispatches the seed URL to the concerned DNS Queues through the URL distributor, and finally the crawling process proceeds. On the other hand if the search for the URL for the specific key word fails (i.e.the keyword has not been visited as yet) the search/insert technique is followed, where the keyword is added to the database and its corresponding URL is added in future when encountered.
The architecture has number of domain specific queues, for the various domains like, .edu, .org, .ac, .com etc. The URLs for these queues are sent to the respective Crawl Workers. This leads to the load sharing by the parallel crawlers on the basis of specific domains.

3.1 Description of the various modules

3.1.1 Crawl Worker

This is the most important module of the entire proposed architecture. The working of the crawl module involves the following steps:

(1) Fetch URL (2) Download Web page (3) Extract URLs (4) Analyze URL (5) Forward URL

Fig.3.2 shows the various modules of the crawl worker. On receiving the seed URL the URL collector forwards it for downloading the Web pages to the downloader. The URL collector stores the downloaded pages in disk; the extracted URLs are stored in Depth URL Buffer temporarily. If Depth URL Buffer is full, all URLs in the buffer are stored in Depth URL File and the buffer becomes clear and sends it to link extractor. The link extractor extracts any URLs in the downloaded pages which are stored in depth URL buffer temporarily .The URL analyzer examines whether the extracted URLs are in syntax error and check duplicate URLs .The extracted URLs are simultaneously submitted to the domain selector, whose task 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.

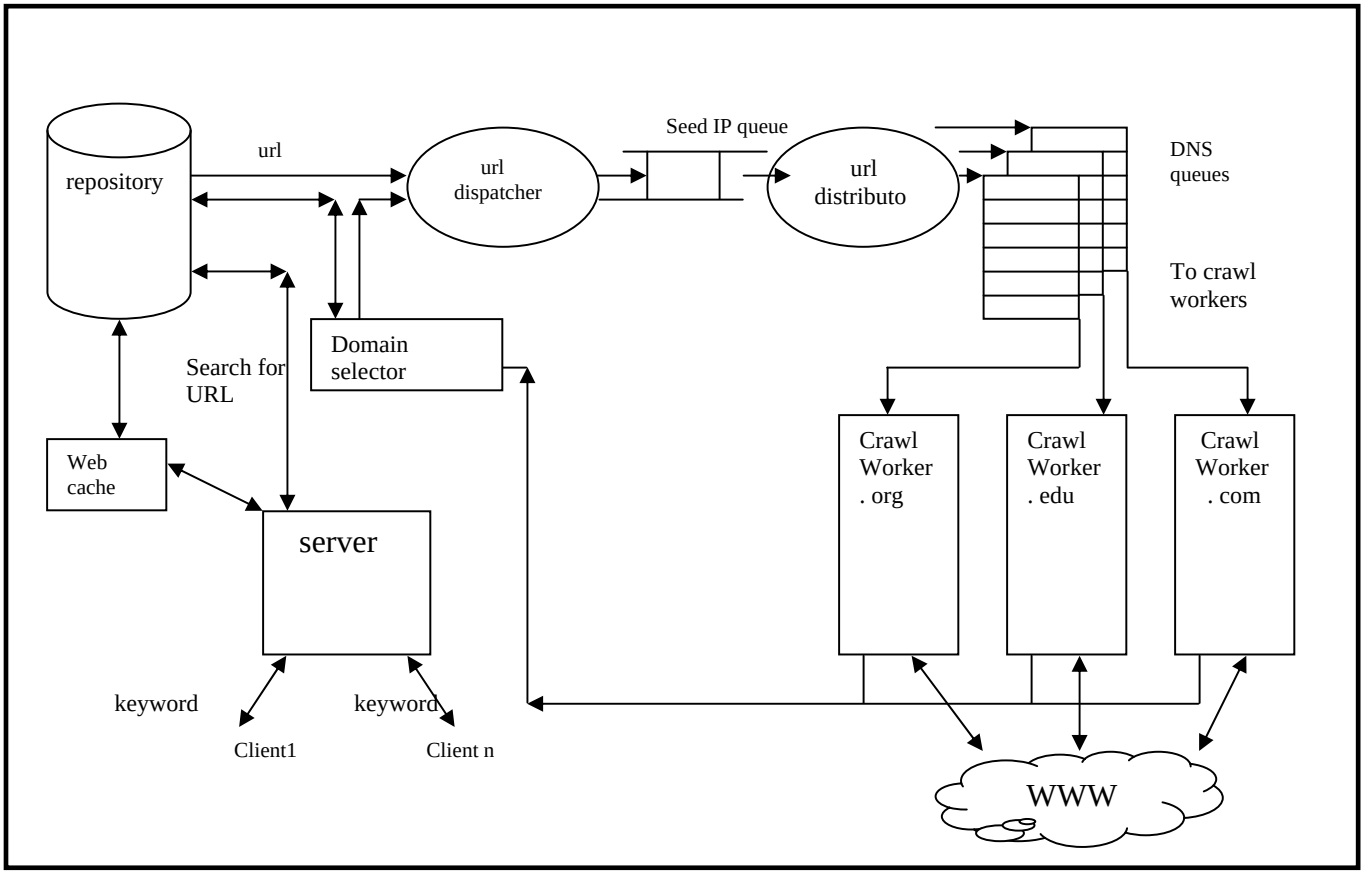


Fig.3.1 Proposed architecture of Domain Specific Parallel Crawler

Downloader

This component takes a URL from URL collector and downloads the corresponding webpage to store it in the local repository. It sends the hungry signal to extract the web addresses from the www and require more URLs for processing. It follows the algorithm given below-

Downloader ()
begin
d*o for ever*
wait (Download WP);
while (empty (LOSBuffer))
begin
extract a URL;
download Webpage;
store webpage in webpage file repository ;
end;
signal (hungry);
end.

The crawl worker helps in load sharing, as DSCrawlers run multiple crawling processes in parallel by multiple machines. Crawling threads download web pages independently without communication between them. Each crawling thread gets a seed URL from the respective DNS Queues.

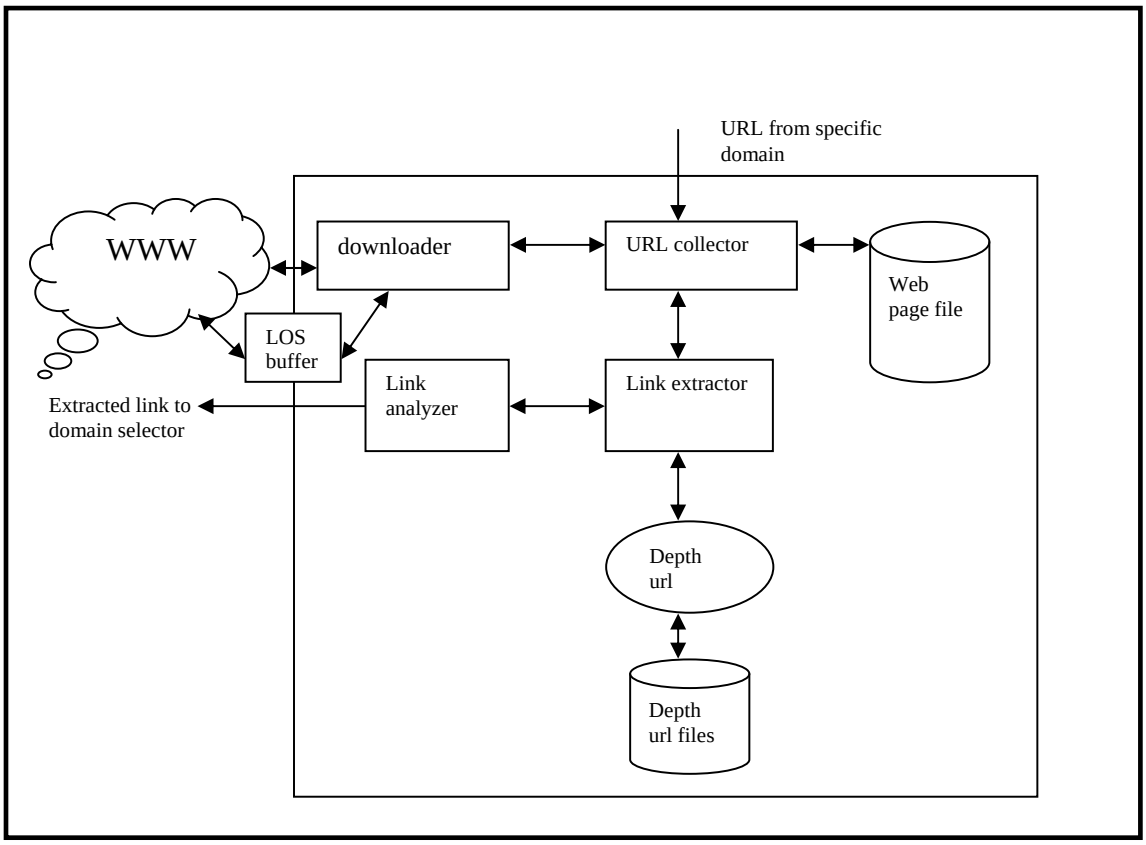


Fig.3.2 Crawl Worker

Link analyzer

Many documents on the web are available under multiple, different URLs. There are also many cases in which documents are mirrored on multiple servers. Both of these effects will cause any web crawler to download the same document contents multiple times. To prevent processing a document more than once, a web crawler may wish to perform URL test to decide if the document has already been processed. Link analyzer (Joo Yong Lee et. al. 2008) makes it possible to check extracted URLs by the link extractor with the depth URL files. If there is similarity in the URL then such URLs are discarded and not further forwarded for processing. Link Analyzer follows the algorithm shown below:

```
Link analyzer()
begin
  while(true)
    if (extracted(URL) = depth url file(URL))
       discard (URL);
    else
   add URL to domain selector
End.
```

3.1.2 URL Dispatcher

Storage and indexing system may simply require a stream of urls from which XML/HTML resources may be downloading. If the user inserts a keyword, then, it is the job of URL dispatcher to receive the concern URL from the index table maintained in the server repository and forward it for crawling. The URL dispatcher read the URL database and fills the URL queue. The URL dispatcher follows the algorithm given below:

```
URL _dispatcher ( )
begin
do forever
    begin
       while (url seed queue not full)
     begin
          read url IP pair from database;
           store it into url IP  queue;
             end
       end
    end.
```

3.1.3 URL Distributor

URL distributor uses the URL distribution algorithm that is illustrated below. URL distributor maintains information on the domains identification of the URL. It gets a seed URL from seed URL queue and distributes the seed URL to the concerned queue of domain specific crawl worker, for further processing. Given a seed URL, the number of downloaded pages or the crawling time depends on the seed URL. The distribution of the load on the crawl workers is likely to be different depending on the frequency of the demand of the domains. URL distributor follows the algorithm shown below-

```
URL Distributor ( )
 While (true)
     if (not empty (seed queue)) then
         src_cw = find domain (URL)
          for each cw in all_crawlworker
              if (not full (DNSqueue_of_src_cw)) then
distribute(src_cw , url) .
```

3.1.4 Domain Selector

The task of the domain selector is to identify the domains of the links extracted by the link extractor and forward the URL in the repository for storage, under the corresponding domain table. On the other hand, the domain selector also forwards the received URL to the URL dispatcher, to repeat the crawling process further.

3.1.5 Repository

Repository, contains the full HTML of every web page. Each page is compressed using z lib/. The choice of compression technique is a trade off between spread and compression ratio. In the repository, the documents are stored one after the other and are prefixed by doc ID, length and URL. The repository requires no other data stretches to be used in order to access it. This helps with data consistency and makes development much easier. The repository stores the database for the keyword and the corresponding URL's as a tree of nodes where each node is a domain. In fact, it is hierarchical structure where in the parent and child is the domains and the sub domains respectively as shown in the Fig 3.3. It may be noted that each node consists of domain name and its corresponding URLs .The database is updated for future use by adding new URL. The keyword inputted by the client is searched in the database, which is the is hierarchical structure and the suitable searching technique (breadth first) (Najork, M., et. al., 2001) is used to get the corresponding seed URL. On positive response the, the appropriate URLs are forwarded to the URL Dispatcher.

3.1.6 Web Cache

A web cache stores Web resources in anticipation of future requests. Web caching works because of popularity the most popular resource is, the more likely it is to be requested in the future. The advantages of Web Cache are
1) reduces network bandwidth usage, which can save money for both consumer and the creator.
2) lessens user-perceived delay, which increases user perceived value.
3) lightens load on the origin servers , saving hardware and support costs for content providers and providing consumers a shorter response time for no cached resources.

When the request is satisfied by a cache, the content no longer has to travel across the Internet from the origin Web server to the cache, saving bandwidth.

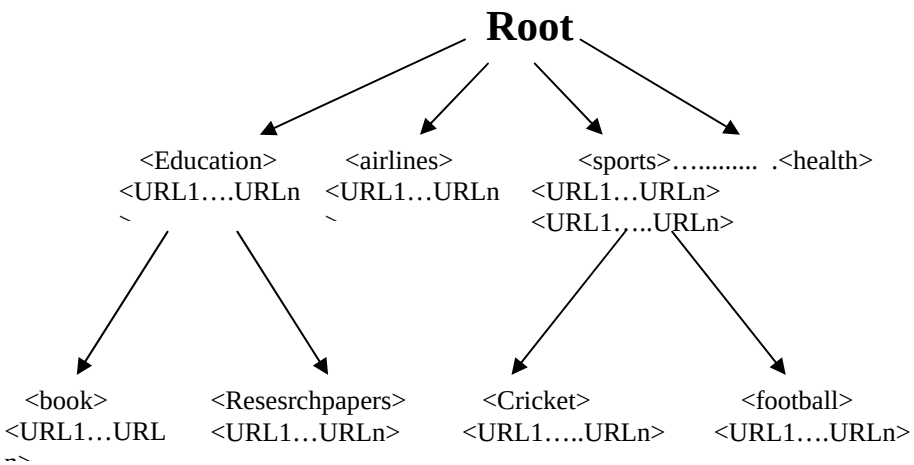


Fig. 3.3 Domain hierarchy in URL Database

4. Example

Step1: The user inputs the keyword through the search engine interface. “COMPUTER NETWORK “is the keyword inputted in the given example. The snapshots of the given example are below.

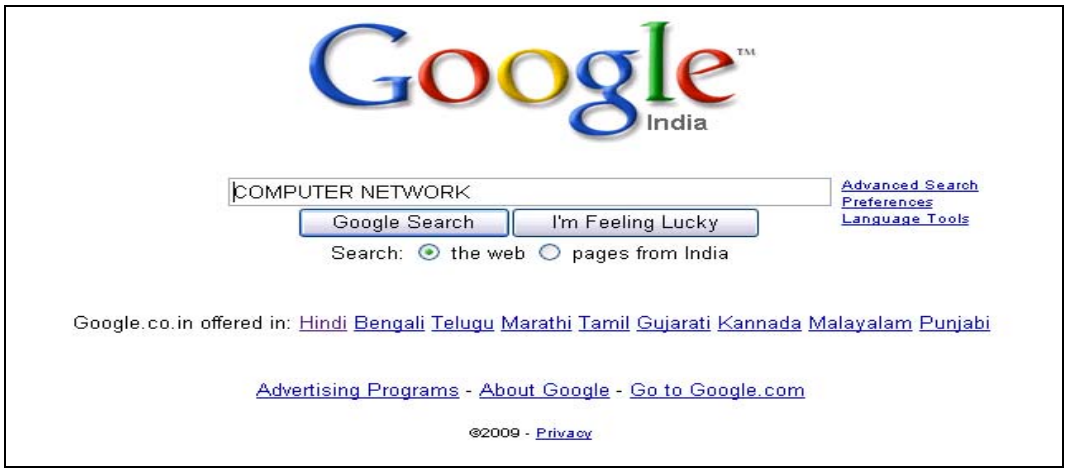
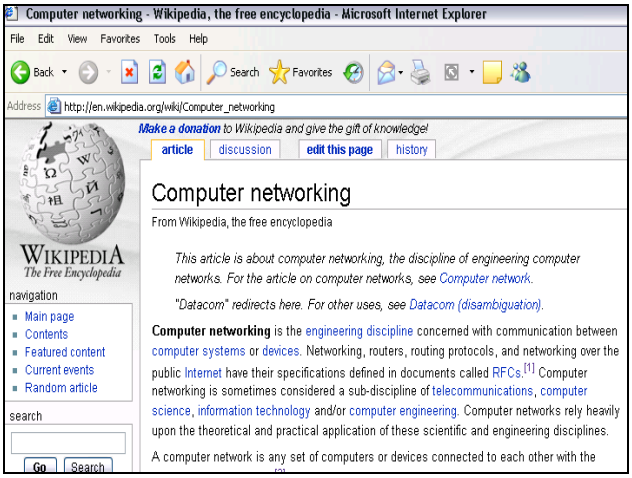


Fig. 4.1 Search Engine interface

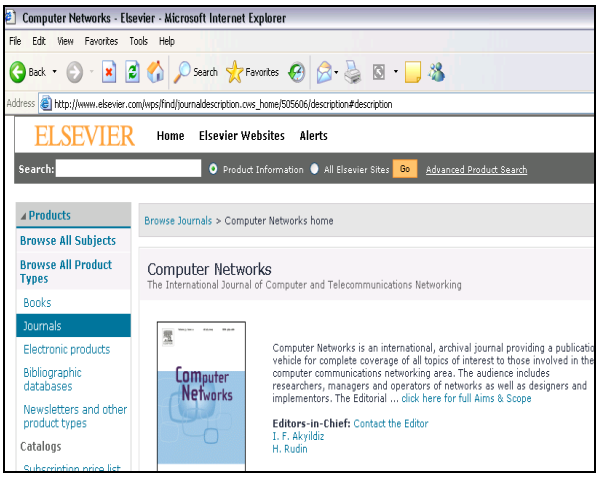
Step2: The keyword search is made in Fig. 4.2 ,which results in the links of different Domains e.g. .com , .org etc. ,which are distributed by URL Distributor to respective DNS queues ,which are further forwarded to different crawl workers.

Fig. 4.2 Searching made for the specific keyword

Step 3. Parallel crawling by different domain specific crawl workers depicted in the Fig.4.3.



.Org Domain Name



.Com Domain Name

.Info Domain Name

.Edu Domain

Fig.4.3 Parallel crawling by domain specific crawlers

Step4. Parallel downloading of the web contents is done by respective crawl workers. Fig. 4.4.

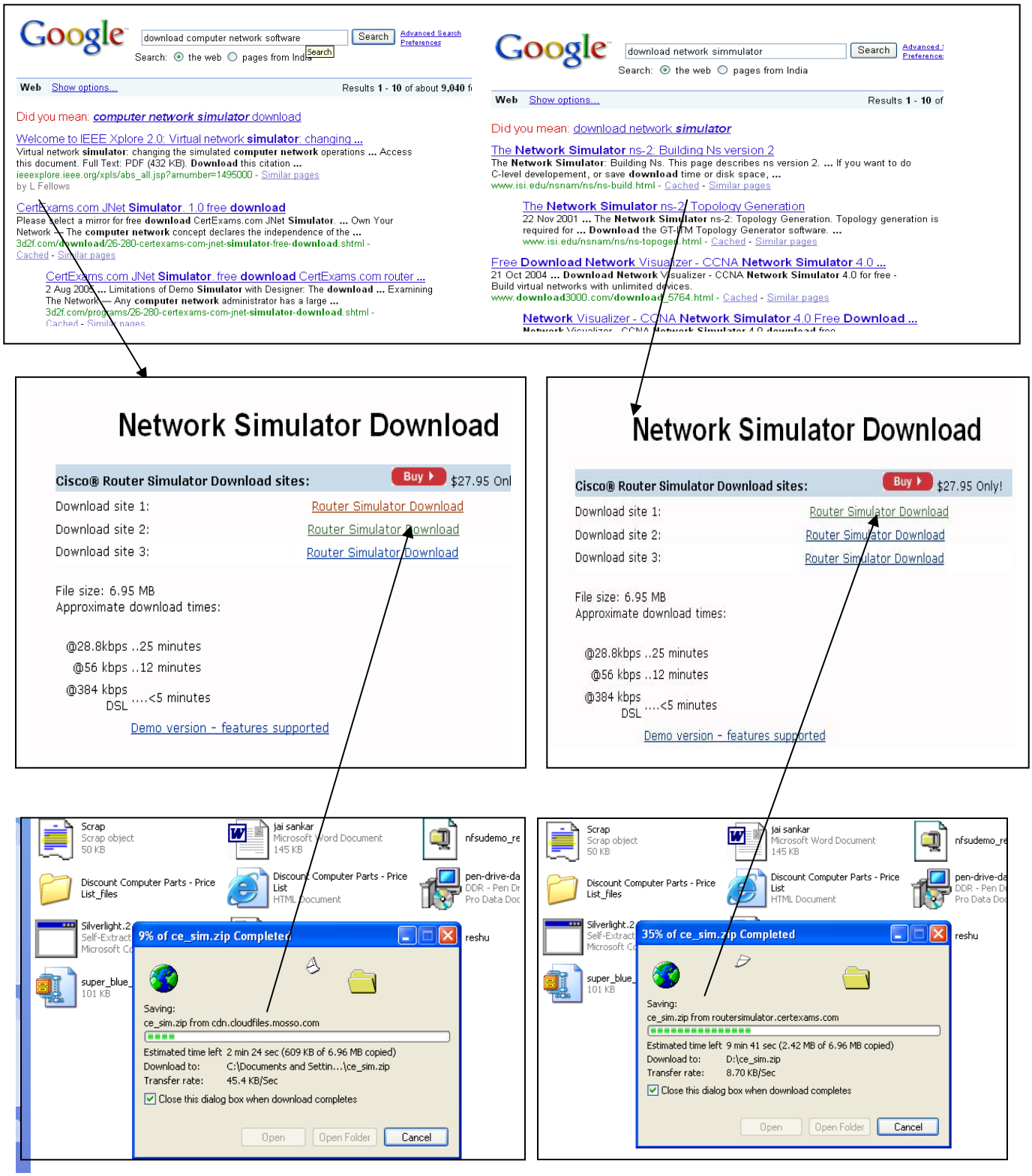


Fig 4.4 Parallel downloading of the web contents different crawl workers

5. Conclusion

This novel architecture is proposed, for building a parallel crawler on the lines domain specific crawling, having the following characteristics-

(1) **Full Distribution:** DSCrawler (Domain Specific Crawler) is distributed over multiple crawling machines (each for specific domain) for better performance. Crawling machines download web pages independently without communication between them.
(2) **Scalability**: Due to the fully distributed architecture of DSCrawler, its performance can be scaled by adding extra machines depending on the bases of the increase of domains, thus manage to handle the rapidly growing Web.
(3) **Load Balancing:** The URLs to be crawled are distributed by the URL Distributor to the particular Domain Specific Queues, thus distributing the crawling to different crawlers which leads the balancing of the crawling load.
(4) **Reliability:** Multiple, independently working crawlers increases the reliability of the whole system, as failure of the single crawl worker will not affect the function remaining crawl workers.

6. Future Work

Though the implementation of the proposed work to enhance the quality of the parallel crawler is the most vital task in the future work. Several interesting issues have been identified which need to be addressed by future research before the proposed architecture can be implemented. Complex Queries: We will explore supporting more complex crawl queries besides the link extraction queries. Example of such queries includes distributed page rank computation, computing web site summaries and constructing inverted indexes of the web page. Fault Tolerance: We have not provided any mechanisms to ensure that the crawl is robust in the presence of node failure. Others: Distribution of the URLs to specific crawlers in minimum amount of time, efficient selection of domains of the link extracted from various sites, enhance the overall crawling speed, in terms of retrieval of information and to provide the crawling security.

References

[1] Yadav, D., Sharma, A. K., and Gupta, J. P. 2008. Parallel crawler architecture and web page change detection. *W. Trans. on Comp.* 7, 929-940.
[2] Balamurugan, Newlin, Rajkumar, J.Preethi, 2008, “Design and Implementation of a New Model Web Crawler with Enhanced Reliability”.
[3] Burner, M., 1997. Crawling towards Eternity: Building An Archive of The World Wide Web. In Web Techniques Magazine, Vol. 2, No. 5, 37-40.
[4] Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual Web search engine. In Computer Networks and ISDN Systems, Vol. 30, No.1-7, 107- 117.
[5] Cho, J., Garcia-Molina, H., 2002. Parallel Crawlers. In WWW’02, 11th International World Wide Web Conference.
[6] 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.
[7] Gray, M., 1996. Internet Statistics: Growth and Usage ofthe Web and the Internet, http://www.mit.edu/people/mkgray/net/.
[8] Heydon, A., Najork, M., 1999. Mercator: A scalable,extensible Web crawler. In World Wide Web, Vol. 2,No. 4, 219-229.
[9] Joo Yong Lee, Sang Ho lee,” scrawler: a seed-by-seed parallel web crawler”, school of computing, soongsil university, seoul, korea,2008.
[10] Junghoo Cho and Hector Garcia – Molina, 2000 “The Evolution of the Web and implementation for an incremental crawler” , Prc. of VLDB Conf..
[11] Najork, M., Wiener, J.L., 2001. Breadth-First Search Crawling Yields High-Quality Pages. In WWW’01,10th International World Wide Web Conference.
[12] http://en.wikipedia.org/wiki/Domain-name
[13] Yadav, D., Sharma, A. K., Gupta, J. P., Garg, N., and Mahajan, A. 2007. Architecture for Parallel Crawling and Algorithm for Change Detection in Web Pages. In *Proceedings of the 10th international Conference on information Technology* (December 17 - 20, 2007). ICIT. IEEE Computer Society, Washington, DC, 258-264.
[14] Yadav, D., Sharma, A. K., and Gupta, J. P. 2009. Topical web crawling using weighted anchor text and web page change detection techniques. *WSEAS Trans. Info. Sci. and App.* 6, 2 (Feb. 2009), 263-275.
[15] Shoubin Dong, Xiaofeng Lu and Ling Zhang, A Parallel Crawling Schema Using Dynamic Partition Lecture Notes in Computer Science Volume 3036/2004, pp. 287-294
[16] Chau, D. H., Pandit, S., Wang, S., and Faloutsos, C. 2007. Parallel crawling for online social networks. In *Proceedings of the 16th international Conference on World Wide Web* (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 1283-1284.

[17] Yu, C. and Lin, S. 2008. Parallel Crawling and Capturing for On-Line Auction. In *Proceedings of the IEEE ISI 2008 Paisi, Paccf, and SOCO international Workshops on intelligence and Security informatics* (Taipei, Taiwan, June 17 - 17, 2008). C. C. Yang, H. Chen, M. Chau, K. Chang, S. Lang, P. S. Chen, R. Hsieh, D. Zeng, F. Wang, K. Carley, W. Mao, and J. Zhan, Eds. Lecture Notes In Computer Science, vol. 5075. Springer-Verlag, Berlin, Heidelberg, 455-466

[18] S,Ye,J Lang,F.Wu, Crawling Online Social Graphs *wwwcsif.cs.ucdavis.edu/~yeshao/papers/apweb10.pdf*