ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.6

Design of Parallel Crawler using Multi-Threading Model

Vineet Singhal[1], Varun Kaushik[2]
M.Tech Student[1], Assistant Professor[2]
Department of CS/IT
MIET, Meerut, India

Abstract:
World Wide Web, is internet client-server architecture, it provides full automated mechanism to server for serving information available on internet. We draw a system that find out links in document. Due to extremely large nature of the pages presented on web, search engines depend upon crawler for collection of pages. A crawler follows hyperlinks presented in documents to download and store the pages in database of search engine. The search engine indexes the pages for later on manipulation of user queries. Web has more than 400 million pages and is growing in the tune of one million pages per day. Such enormous growth and flux necessitates the creation of highly efficient crawling system. A Crawler is a program that downloads and stores web pages, often for Web Search Engines. It is obvious that deploying a single crawler to retrieve the whole or significant portion of web is very time –consuming and difficult to complete. Therefore, many search engines often run multiple crawlers in parallel to fulfill tasks, in order to achieve a maximum download rate. In this paper, work explores as how to download relevant web pages by parallel crawler without redundancy according to user query. As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. We refer to this type of crawler as a parallel crawler.

1. INTRODUCTION

World Wide Web is internet client-server architecture; it provides full automated mechanism to server for serving information available on internet. We draw a system that find out links in document that is hyperlink in other document, such in form of text or images. Due to extremely large nature of the pages presented on web, search engines depend upon crawler for collection of pages.

A crawler follows hyperlinks presented in documents to download and store the pages in database of search engine. The search engine indexes the pages for later on manipulation of user queries. Web has more than 400 million pages and is growing in the tune of one million pages per day. Such enormous growth and flux necessitates the creation of highly efficient crawling system. Internet Archieve uses multiple crawler process to crawl the web. Each crawler process is single threaded which takes a list of seed URLs and fetches page in parallel.

The links are extracted from downloaded documents and placed in different data structures. A Crawler is a program that downloads and stores web pages, often for Web Search Engines. It is obvious that deploying a single crawler to retrieve the whole or significant portion of web is very time –consuming and difficult to complete. Therefore, many search engines often run multiple crawlers in parallel to fulfill tasks, in order to achieve a maximum download rate.

Competition among parallel crawling results in redundant crawling, wasted resources, and less-than timely discovery. In intuitive, the parallel crawler has many advantages compared to single process crawler, such as scalability, network load distribution and reduction.

2. GENERAL ARCHITECTURE FOR PROPOSED PARALLEL CRAWLER

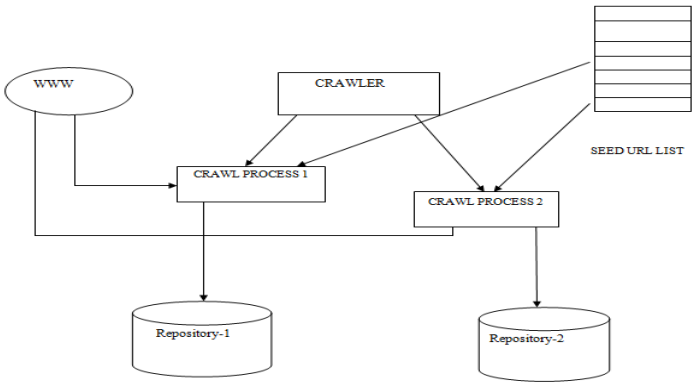


Figure.1. General Architecture for Parallel Crawler

From the crawler main process two crawling threads are created crawl process1 and crawl process2 respectively, list of url's presented in the URL list is divided and equally submitted to the two crawling processes simultaneously, then the list of url's collected by the two sub processes separately in repository-1 and repository-2 respectively are again send to the main URL list. And also the pages are collected in the respective repositories.

2.1. DETAILED DESCRIPTION OF MODULES OF PROPOSED WORK PLAN

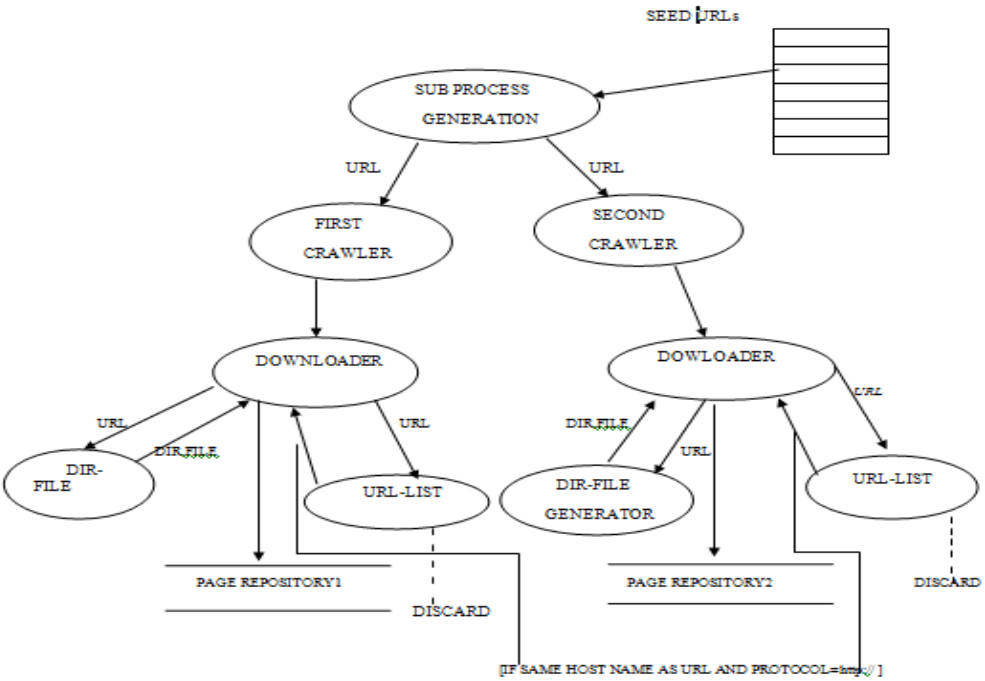


Figure.2. DFD for Proposed Methodology

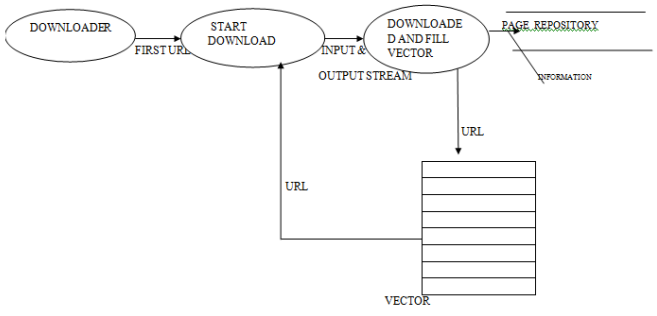
Module -Description

1: SUB PROCESS GENERATION:

Sub Process Generation generates crawler according to augmented urls in Seed URLs mention. In this module, Each crawler process is single threaded which takes a list of seed URLs and fetches page in parallel. The links are extracted from downloaded documents and placed in different data structures.

2: DOWNLOADER:

Downloader download such types links from relvant documents that is needed according to passes argument urls in Seed URLs, that also hyperlinks to other relevant documents.



The algorithm used by the Downloader is below as:

```
Downloader()
        {
    Start Download()
        }
startDownload()
    {
fdir? getDirectory()
if(fdir not exists)
        mkdirs()
ffile? getFile()
if(ffile not exists)
        ffile? index.html
else
        ffile? filename
list Of URL? Download And Fill Vector (Input Stream,Output Stream)
while(listOfURL not empty)
        start Download (each URL in list Of URL)
    }
Download And Fill Vector (Input Stream, Output Stream)
{
while(characters read from ffile !=EOF)
if(hyperlink)
        v? hyperlink
else
        write(characters)
return form vector Of URLs(v)
    }
formvectorOfURLs(v)
        {
Returnget URLs()
}
```

3. DIR-FILE GENERATOR

In Dir-File Generator takes urls as input and generate Directories and Files, by which stored information on client machine such as these exist on remote machine. using this type of parsing in sense of directories and files more benefit due to stored gathered information in proper ordering of paths of documented pages. This module checks whether the file mentioned in the URL is a directory or a file, if no file is mentioned in the URL it will set the file as "index.html". The algorithm used by the dir-file is shown as below:

```
Get File()
{
return file
}
Get Directory()
{
                returndir
}
Parse (url)
{
tmpstr? file in the url
        if(tmpstrends with "/")
                dir? substring of tmpstr from 0 to
length[tmpstr]
else
                idx? Last Index Of '/' in tmpstr
                file? substring of tmpstr from idx+1 to
length[tmpstr]}
```

4. URL LIST

Under URL LIST, check a url which is comes to downloader for downloading your respective web pages that is match or not match according to host name and protocol=http:// of augmented seed url that is passed in List of Seed URLs. If match according to above criteria then allowed for downloading otherwise downloading of that url is discarded by downloader. In this module also check if any web pages that is already downloaded by downloader then discarded downloading. So under this module reduced redundancy of downloading of same pages by downloader. This module fetches a URL from vector and provided it to the Downloader module. The algorithm for the URL LIST is implemented by the function getURLs (). The algorithm used by the URL LIST is shown as below:

```
Get URLs()
{
Size Of Vec? Size [list Of URLs]
for i?1 to size Of Vec
                add To List(url)
returnrlist Of URLs
}
Add To List (url)
{
If(hostname equals url host)
                If(URLs of Downloader not contain url)
                        addurl to rlist Of URLs}
```

5. PAGE REPOSITORY

In Page Repository, stored download web pages that is downloaded by respective downloader.

6. CONCLUSION AND SCOPE OF FUTURE WORK

A Crawler is a program that downloads and stores web pages, often for Web Search Engines. It is obvious that deploying a single crawler to retrieve the whole or significant portion of web is very time –consuming and difficult to complete. Therefore, many search engines often run multiple crawlers in parallel to fulfill tasks, in order to achieve a maximum download rate. This paper, explores an efficient technique to download relevant web pages by parallel crawler without redundancy according to user query. As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. We refer to this type of crawler as a parallel crawler. This system also checks that the same URL not submitted to the parallel crawler running simultaneously. The proposed system increases the retrieval of URL's. In future the system is enhanced by first checking the load of each crawling process by applying some load checking algorithm and after that submit the URL to the crawling process having less amount of load.

7. REFERENCES

[1]. Cho, J., Garcia-Molina, H. 2002.Parallel Crawlers.In WWW'02, 11th International World Wide Web Conference.

[2]. Joo Yong Lee, Sang Ho lee," scrawler: a seed-by-seed parallel web crawler", school of computing, soongsil university, seoul, korea,2008.

[3]. 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.

[4]. D.Yadav, A.K. Sharma and J.P.Gupta, "Change Detection in Web pages", IEEE Proceeding of 10th International Conference on IT, Dec 17-20, 07, Rourkela (India). ISBN: 0-7695-3068-0, Page(s) 265-270

[5]. A.K.Sharma and Komal Kumar Bhatia,A Framework for Domain-Specific Interface Mapper (DSIM)

[6]. A. K. Sharma, Komal Kumar Bhatia: "Automated Discovery of Task Oriented Search Interfaces through Augmented Hypertext Documents" accepted at First International Conference on Web Engineering & Application (ICWA2006).

[7]. Dustin Boswell, "Distributed High-performance Web Crawlers:A Survey of the State of the Art"

[8]. A. K. Sharma, J. P. Gupta, "An Architecture of Electronic Commerce on the Internet", accepted for the publication in the fourth coming issue of Journal of Continuing Engineering Education, Roorkee, Jan 2003

[9]. Dinesh Sharma, A.K. Sharma, Komal Kumar Bhatia, "Web crawlers: a review", NCTC-2007

[10]. Dinesh Sharma, A.K. Sharma, Komal Kumar Bhatia," Search engines: a comparative review",NGCIS-2007

[11]. http://www.cs.utah.edu/~juliana/pub/webdb 2005. pdf

[12].http://en.wikipedia.org/wiki/Deep_web

[13]. www.invisibleweb.com

[14]. Alexandros Ntoulas Petros Zerfos Junghoo Cho, "Downloading Hidden Web Content", UCLA

[15]. A. Bergholz and B. Chidlovskii. Crawling for Domain-Specific Hidden Web Resources. In Proceedings of WISE, pages 125–133, 2003

[16]. http://en.wikipedia.org/wiki/parsing

[17]. http://www.brightplanet.com/imagesstories.pdf/deepweb whitepaper.pdf

[18]. http://amazon.com

[19]. www.cs.utah.edu/~juliana/pub/freire-sbbd2004.pdf

[20]. http://www2007.org/papers/paper429.pdf

[21]. http://www.lib.berkeley.edu/TeachingLib/Guides/Internet /InvisibleWeb.html

[22]. http://www.weblens.org/invisible.html

[23]. http://websearch.about.com/od/invisibleweb/a/invisible_ web.htm

[24]. http://www.searchengineshowdown.com/features/av/r eview. html

[25]. http://harvest.sourceforge.net/harvest/doc/index.html

[26]. www.searchengineshowdown.com/features/google/review. html

[27]. http://infolab.stanford.edu/~backrub/google.html#hits

[28]. http://www.asis.org/annual-96/Electronic Proceedings/ chu.html

[29]. http://daphne.palomar.edu/TGSEARCH/

[30]. The Overview of Web Search Engines written by Sunny Lam