ISSN:0975-9646
Ela Kumaret al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1700-1705

CurlCrawler:A framework of CurlCrawler and Cached Database for crawling the web with thumb.

Dr Ela Kumar[#],Ashok Kumar[$]
 # School of Information and Communication Technology, Gautam Buddha University, Greater Noida
$ AIM & ACT.Banasthali University,Banasthali(Rajasthan.)

***Abstract*-WWW is a vast source of information and search engines are the meshwork to navigate the web for several purposes. It is problematic to identify and ping with graphical frame of mind for the desired information amongst the large set of web pages resulted by the search engine. With further increase in the size of the Internet, the problem grows exponentially. This paper is an endeavor to develop and implement an efficient framework with multi search agents to make search engines inapt to chaffing using curl features of the programming, called CurlCrawler. This work is an implementation experience for use of graphical perception to upright search on the web. Main features of this framework are curl programming, personalization of information, caching and graphical perception.**

Keywords and Phrases-Indexing Agent, Filtering Agent, Interacting Agent, Seolinktool, Thumb, Whois, CachedDatabase, IECapture, Searchcon, Main_spider, apnasearchdb.

1. INTRODUCTION

Information is a vital role playing versatile thing from availability at church level to web through trends of books. WWW is now the prominent and up-to-date huge repository of information available to everyone, everywhere and every time. Moreover information on web is navigated using search engines like AltaVista, WebCrawler, Hot Boat etc[1].Amount of information on the web is growing at very fast and owing to this search engine's optimized functioning is always a thrust area that makes it as a buzz word of research area. At the ground level, a Search Engine employs Crawlers, which traverse the web by downloading the documents and following links from page to page. Since, Crawlers gather data for indexing; these form the most important part of a Search Engine. The aim of this paper is to raffle a framework, which will elevate Crawler's dexterity to surmount the way the Internet can be used to snag more and more information and services [2,3,4].

This paper presents design and implementation of CurlCrawler, featured with locally resource utilization capacity to deliver more personalized, graphical and cached driven information from the web. This crawler is destine to present a framework, which will convince the chaffing experience while searching on the Internet.[1]

2. RELATED WORK

Although information about the functioning issues of professional search engines is not available publicly and if available then it is only up to concept level [13]. This is a business driven scenario owing to which we find out only about basic modules of search engine functioning and these essential modules are (see Fig.1)[9,10].

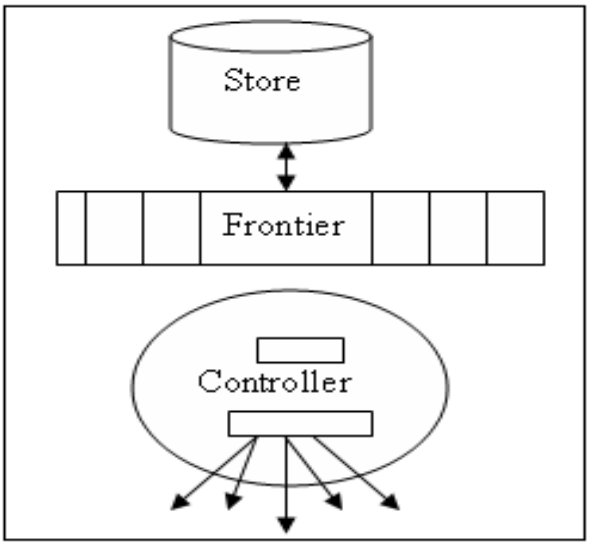

Fig.1: Components of Information Retrieval System

Store: It stores the crawled pages. Its main functions are:

- To check whether a page has already been created
- To store the contents of crawled pages
- To keep track of some relevant information about its stored pages.

Frontier: This component deals with the retrieval of new pages. Its main functions are:

- To keep track of the URLs that have to be crawled by the agent,
- To actually fetch the content of the URL to be crawled
- To parse the retrieved URL.

Controller: It overseas all the communications between agents and works as a reliable crash failure detector. The reliability refers to the fact that a crashed agent will eventually be distrusted by every active agent. It also determines through delegation function as to which agent is responsible for each single URL. The delegation function also partitions the web domain in such a way that every running agent is assigned approximately the same number of URLs.

3. SOFTWARE ARCHITECTURE

Software Architecture is the set of structures needed to reason about the system, which encompasses the set of significant decisions about the organization of the developed framework including the selection of the structural elements and their interfaces by which the system is composed and an architectural style that guides this organization. Software architecture of developed framework employs different software elements as described below[6,8].

3.1 Architecture of CurlCrawler.
3.2 Cached database architecture.
3.3 Interacting agent architecture.

3.1 Architecture of CurlCrawler

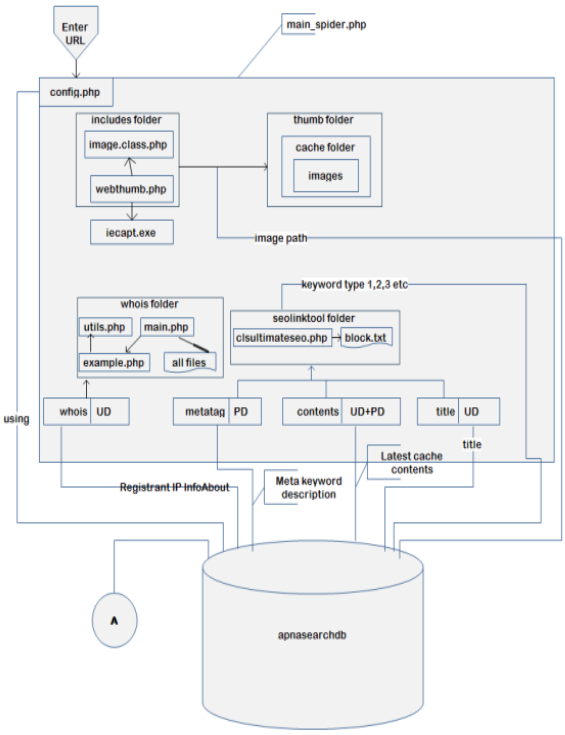

Fig.3.1: Architecture of CurlCrawler

Main_spider: is an agent that crawls the web for information of URL of the website, Title of the website, Meta keyword used up to three or four levels for website, Meta keyword description used up to three or four levels for website, Website keywords with one word pattern, Website keywords with two word pattern, Website keywords with three word pattern, Website context, Links on website, Links visited on website, Content to be cached, Date and time on which cached by, Information about hosting server, Information of registrant, Additional information about website owner, Additional information about website, Website link filed anywhere else in our database, Total number of visitors, Website created on, Website updated on and already crawled or not. All of this info is indexed and stored to database using indexing software agent deployed. This agent collects and creates an indexed database using the following modules[5,7].

Modules	Description
Config	is responsible for database connectivity.
Includes	employs imageclass, webthumb and IEcapture sub modules that generate thumb for given URL's home page and sends this to Thumb module to save this as an image in Cache folder.
Thumb	is responsible to send thumb for that URL as an image from cache folder to database for indexing and storage.
SEOLinktool	uses ClsUltimateSEO and Block sub modules to fetch title, meta keyword description, cache contents, content latest and keyword with different patterns.
WhoIs	uses Utils, Main and Example sub modules to fetch whois information like registrant, server IP information about for that URL. This also sends fetched information to database module for storage and indexing.

3.2 Cached database architecture

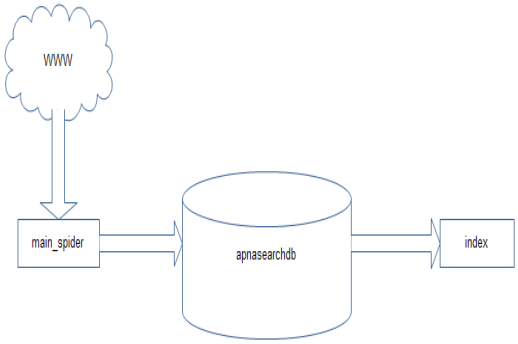

Fig. 3.2: Architecture of Cached Database

Index: is a filtering module that provides user perception and interest to be used to fetch result from database server.

3.3 Interacting agent architecture

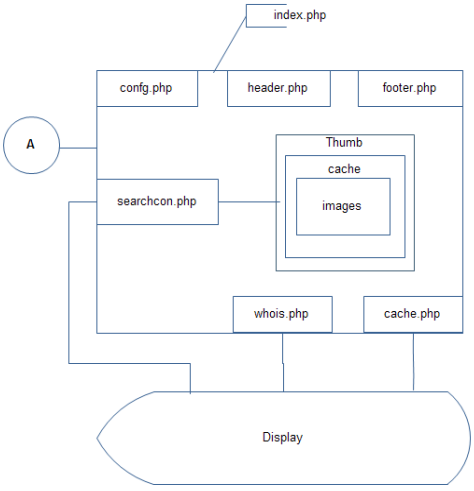

Fig.3.3:Architecture of Interacting Agent

Interacting Agent: gets keyword(s) to search indexed database and expel result page using the following sub modules:

Modules	Description
Config	is responsible for database connectivity.
Header	creates header of the result page or user interacting page
Footer	creates footer of the result page or user interacting page.
WhoIs	Fetches database to read all personalized information stored in context of keyword.
Cache	Fetches database to read cache copy of website stored in context of keyword searched.
Searchcon	key module that fetches database for rest of the information specially image using thumb and cache sub modules in context of keyword entered.
Display	is responsible to merge result and layout generated by above explained modules.

4. PERFORMANCE

An estimated and approximate performance analysis can be done to compare the conventional search strategies with the developed one. With the increase in the availability of web pages on the Internet, the major problem faced by the present search engine is difficulty in information retrieval [11]. It is problematic to identify the desired information amongst the large set of web pages resulted by the search engine. With further increase in the size of the Internet, the problem grows exponentially (see Fig.4.0). The number of web pages given as the result of a user initiated will definitely be more.

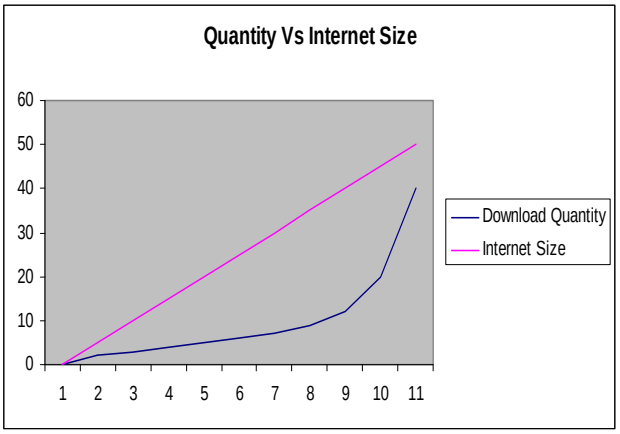

Fig.4.0: Download Quantity vs. Internet Size.

This increase in the quantity on one hand, leads to decrease in the quality (see Fig.4.01) on the other. The framework given in this work, effectively takes into consideration the above mentioned issues. Being a context driven search strategy, use of local resources i.e. curl programming features, reduced chaffing owing to more information like thumb, caching the framework is a key step for search mechanism with less degree of chaffing.

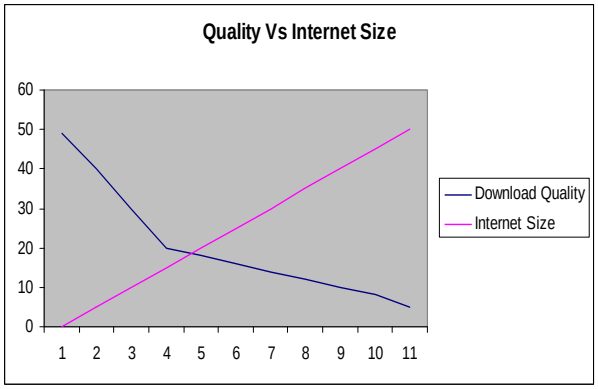

Fig.4.01: Download Quality vs. Internet Size

In terms of performance parameters like quantity, quality, relevance with the keyword searched and the network traffic; developed framework holds an edge above the conventional search strategies. The results are more pertinent to the user's interest owing to more focused, relevant, personalized, cached and graphical.

4.1 Experimental Screenshots

A series of user interfaces of developed framework with deployed CurlCrawler(see Fig. 4.1, 4.2, 4.3, 4.4) while rendering for a keyword "job" is shown below:

Fig.4.1: Home Interface

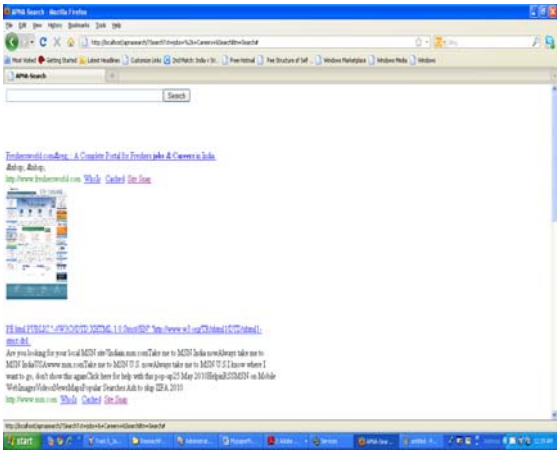

Fig.4.2:Thumb Created Result

Fig.4.3:WhoIs Info Result

Fig.4.4:Cache Result

4.2 Analysis

This framework is running on an acer machine, a workstation with 685MHz processor, 12 GB of RAM,840 GB of local disk, 100 Mbit/sec Speed Internet, Windows Server 2003 and xampp 1.7.3.

In this paper, experimental statics are presented of 9 days only owing to compare with other existing search systems like Google, about this request issued are published in literature. The Google crawler is reported to have issued 26 millions HTTP requests over 9 days i.e. on an average 33.5 docs/sec and 200KB/sec[14,15]. Performance of any information retrieval system can be analyzed using parameters like coverage and user perception that are presented below:

4.2.1 Coverage

Coverage of a search engine points towards a search engine's crawl speed and index size. In case of developed framework, CurlCrawler made 68.5 millions HTTP requests in 9 days, achieving an average download rate of 99.121 docs/sec and 1488.59 KB/sec. Hence, this work with local resource utilization is a considerable optimization mark and represented as below (seeFig.4.2.1):

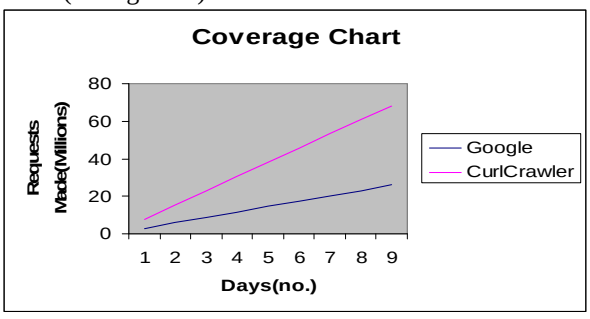

Fig.4.2.1 Coverage Chart

4.2.2 User Perception

User perception points towards user experience with developed framework. In this work, key points towards user perception are:

GUI perception

Out of 68.5 million requests made, 1.36 millions requests do not return thumb i.e.1.985% and 0.62 millions requests return a thumb that is not clear up to the identifying mark i.e. 0.905%(see Fig.4.2.2.0).

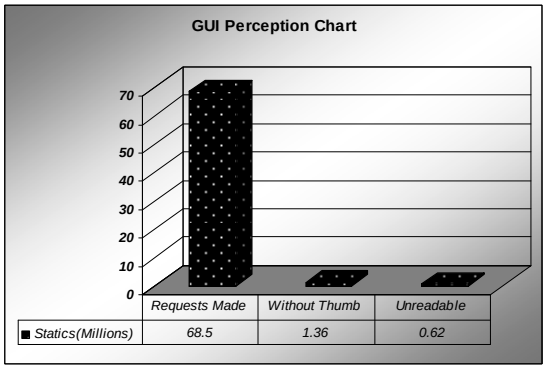

Fig.4.2.2.0 GUI Perception Chart

Personalization degree

Out of 68.5 million requests made , 0.14 millions requests do not return personalization of information like registrant, hosting info etci.e.0.204%.(see Fig.4.2.2.1).

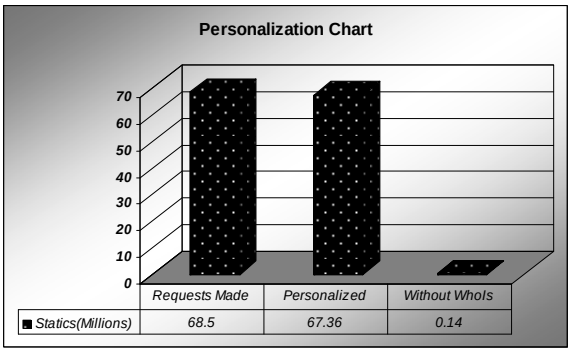

Fig.4.2.2.1 Personalization Chart

Hence, these are the wrinkled points of this work that were not expected to be happened.

5. CONCLUSION

This framework renders the web for additional information like thumb, cache, registrant and higher degree of context to provide more interesting perception from users interacting with. This is a part of ongoing research work, to utilize advance features of programming in crawling the web up to maximum extent. Owing to the lengthy size of coding work, this is not possible to present coding or technical details of all the modules of developed framework. But work is incomplete without functioning details of the basic modules i.e. index module and main_spider module.

5.1 Index

Basic technical details like pseudo code and data structures are given below:

Individual Data Structures Used:

Name	Type	Usage
SearchFrm	Form	To create result page
SearchTxt	Textbox	To enter query
SearchBtn	Submit Button	To search result from database
Type	String	To store corresponding Id of event occurred
Cache	Link Button	To print cache result
WhoIs	Link Button	To print personalized result
Thumb	Link Button	To display thumb result

Pseudo code:

```
Create header;
Create form with one textbox, one submit button, one cache
and one thumb link button;
if(type == 'whois')
{
        call functions of module 'whois.php';
}
if(type == 'cache')
{
        call functions of module 'cache.php';
}
if(type == 'searchbtn')
{
        call functions of module 'searchcon.php';
}
Create footer;
```

5.2 Main_spider

Basic technical details like pseudo code and data structures are given below:

Individual Data Structures Used:

Name	Type	Usage
url	String	To store url value
responseTitle	String	To store fetched title corresponding to url value
metaContent	String	To store fetched meta tags corresponding to url value
urlContents	String	To store fetched url contents corresponding to url value
keyContent	String	To store fetched keywords corresponding to url value
whoIsInfo	String	To store fetched whois information corresponding to url value
registrantInfo	String	To store fetched registrant information corresponding to url value
thumbName	String	To store path of created thumb corresponding to url value

Common Data Structures Used:

Name	Type	Usage	Degree
web_contents	Table	To store Complete information	22

Pseudo code:

```
read url;
getAllDetailsInDb(url);
function getAllDetailsInDb(url)
{
        responseTitle = getTitle(url);
        metaContent = get_meta_tags(url);
        urlContents = getURLcontents(url);
        if(count(trim(urlContents)) <= 200)
    {
        urlContents = file_get_contents(url);
        stripContents = urlContents;
        }
        stripContents = strip_tags(urlContents);
        keyContent =
fetchKeywordContents(url,stripContents);
        oneWordTexts = "";
        foreach(keyContent["_1"])
    {
                oneWordTexts =Val;
        }
        twoWordTexts = "";
        foreach(keyContent["_2"])
    {
                twoWordTexts=val;
        }
        $threeWordTexts = "";
        foreach($keyContent["_3"])
    {
                threeWordTexts =Val;
        }
        whoIsInfo = getWhoIsInfo(url);
        thumbName = makeThumbnel(url);
        whoIsNServer = "";
        foreach(whoIsInfo['regrinfo']['domain']['nserver'])
    {
                whoIsNServer=value;
        }
        registrantInfo =whoIsInfo['regyinfo']['registrar'];

        whoIsFullInfo = "";
        foreach(whoIsInfo['rawdata']=> value)
    {whoIsFullInfo=value;}
        parsedDate = date("Y-m-d H:i:s");
        rsAlreadyQuery = mysql_query(AlreadyQuery);
        if(rowAlreadyQuery =
mysql_fetch_assoc(rsAlreadyQuery))
    {Update existing record;}
    else
    {Insert new record;}
}
```

Finally, the complete framework along with implementation details of various agents used is discussed. A crawler executing in a Multi-Agent environment is designed and developed to expel a search that is more focused, relevant, personalized, cached and GUI driven. An extension to the developed framework is also going on that uses an additional agent named Learner Agent, which could observe, analyze and imitate the user. It could formulate the right set of keywords and proactively trigger a new query on its behalf[12].

6. REFERENCES

[1]. Agichtein et al.(2006), "Learning user interaction models for predicting web search result preferences.", SIGIR'06.

[2]. Google (2008), Webmaster Guidelines.http:// www.google. com/support/

[3]. Yahoo Search Content Quality Guidelines (2008).http:// help. yahoo. com/l/us/yahoo/search/basics/basics-18.html

[4]. Kobayashi, M. and Takeda, K. (2000). "Information retrieval on the web". ACM Computing Surveys (ACM Press)

[5]. Cho,Junghoo; Hector Garcia-Molina (2002). "Parallel cawlers". Proceedings of the 11th international conference on World Wide Web.Honolulu, Hawaii, USA: ACM.pp. 124–135. doi:10.1145/511446.511464. ISBN 1-58113-449-5.

[6]. Zeinalipour-Yazti, D. and Dikaiakos, M. D. (2002). Design and mplementation of a distributed crawler and filtering processor. In Proceedings of the Fifth Next Generation Information Technologies and Systems (NGITS)

[7]. Pant, Gautam; Srinivasan, Padmini; Menczer, Filippo (2004). "Crawling the Web". in Levene, Mark; Poulovassilis, Alexandra. Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer. pp. 153–178. ISBN 9783540406761.

[8]. Shkapenyuk, V. and Suel, T. (2002). Design and implementation of a high performance distributed web crawler. In Proceedings of the 18th International Conference on Data Engineering (ICDE), pages 357-368, San Jose, California. IEEE CS Press.

[9]. Edwards, J., McCurley, K. S., and Tomlin, J. A. (2001). "An daptive model for optimizing performance of an incremental web crawler". In Proceedings of the Tenth Conference on World Wide Web (Hong Kong: Elsevier Science)

[10].Cho, J.; Garcia-Molina, H.; Page, L. (1998-04). "Efficient Crawling Through URL Ordering". Seventh International World-Wide Web Conference. Brisbane, Australia.

[11].Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused crawling: a new approach to topic-specific web resource discovery. Computer Networks, 31(11–16):1623–1640.

[12].Shestakov, Denis (2008). Search Interfaces on the Web: Querying and Characterizing. TUCS Doctoral Dissertations 104, University of Turku.

[13].AltaVista Search Contents http://www. infotoday. com/ searcher/may01/liddy.htm

[14].Sergey Brin and Lawrence Page. The anatomy of a large-scale pertextual Web search engine. In Proceedings of the Seventh nternational World Wide Web Conference, pages 107--117, April 1998.

[15].Z.Smith. The Truth About the Web: Crawling towards Eternity. Web Techniques Magazine, 2(5), May 1997.