International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 1, January- 2013

Curlcrawler Optimization: A Framework For Crawling The Web With URL Tracking And Canonicalization.

Ashok Kumar#
#Banasthali University,Banasthali(Raj.)-India

Dr. Saurabh Mukherjee[$]
Banasthali University,Banasthali(Raj.)-India

Manisha Garhwal[@]
W3technosoft, Jaipur-302018, India

Abstract

Information is a vital role playing versatile thing from availability at church level to web through trends of books. WWW is now the huge, exposed, up-to-date, interoperable and dynamic repository of information available to everyone, everywhere and every time. In addition to the size of information available on the web its scheme, authority, dynamism, appearance and interoperability are the attributes that are growing and adopted exponentially[7,10]. These attributes are the directing one to coin a new term web 2.0 that is an evolution of web from its embryo. Search engines are the striking one to sail the web for several purposes because moreover information on the web is voyaged using search engines like AltaVista, WebCrawler, Hot Boat etc. Owing to the directing factors of its ever -growing exponential growth with the availability of endless pool of information, optimization of its design blueprint is the thrust arena of engineering endeavor.

This paper is an experimental strives to develop and implement an extended framework with extended architecture to make search engines more efficient using local resource utilization features of the programming. This work is an implementation experience for use of focused and path oriented approach to provide a cross featured framework for search engines with human powered approach. In addition to curl programming, personalization of information, caching and graphical perception, main features of this framework are cross platform, cross architecture, url tracking, focused, path oriented, human powered and url canonicalization[7,21.

The first part of the paper covers related work that has been done mostly in the field of general search engine in over ongoing research project for crawling the web. The second part defines architecture and functioning of developed framework and compares it to search engine optimization for web pages. The third part provides an overview and critical analysis of developed framework like experimental results, pseudo code, data structure etc.

Keywords and Phrases: Path String, Indexing Agent, Filtering Agent, Presentation Module, Seolinktool, Thumb, Whois, CachedDatabase, IECapture, Searchcon.

1. Introduction

The aim of this paper is to raffle a framework, which will elevate search engine's dexterity to surmount the way the Internet can be used to snag more and more information and services even with the directing factors of its exponentially growing endless pool of information. Finally, an age has come, where information has become an instrument, a tool that can be used to solve and mean to many problems. Moreover information on web is navigated using search engines like AltaVista, WebCrawler, Hot Boat etc [1,7] that deploys a software module called crawler. Optimization of search engine functioning is a raptorial field to address a state of fast growing rate, nature and state of amount of information on the web. A typical web crawler starts by parsing a specified web page and noting any hypertext links and other information relevant to result optimization on that page that point to other web pages. The crawler then parses those pages for new links, and so on, recursively. All the crawler really does is to automate the process of following links [10].

This is the basic concept behind implementing web crawler, but implementing this concept is not merely a bunch of programming. Large volume and other directing factors on web pages are the important characteristics of the web that generate a scenario in which efficient web crawling is very difficult. Another problem of dynamic world is that web pages on the Internet change very frequently, as a result, by the time the crawler is downloading the last page from a site, the page may change or a new page has been placed to the site or same information is authored by more than on url. First problem is addressed by devising the concept of url tracking and second one is addressed with url canonicalization[11]. Url tracking is the process to implement the indexed database created by index agent of the crawler as a vertex while starting from seed url and finally a graph having internal links as an in-degree and external links as an out-degree of the current vertex. Url tracking can be applied in both direction that is forward tracking and backward tracking. Url canonicalization is the normalization process of transforming URL strings into canonical form. After normalization, identically transformed URLs are regarded as equivalent URLs. Basically, the URL normalization determines whether two URLs are equivalent prior to access to the corresponding web pages[7,21].

The difficulties in implementing efficient web crawler clearly state that crawling is not the only function of search engine to be optimized up to next extent but guidelines for reduced chaffing result as well.

The growing popularity of search engine has emerged as a handy tool for information retrieval and a preferred media for advertising. The benefit of getting top rankings from popular search engines includes online presence and sales boost [11] resulting the strategy of pay per click. Listing in top search results is desirable as most of the users conclude their search in one or two pages of search results or pay per click area [6, 10]. This behavior of user has necessitated a need for ranking friendly crawler and webmasters to modify their web pages with a goal to improve their search ranks.

This paper presents extended design and implementation of optimizing CurlCrawler, featured with cross platform, cross architecture, focused, path oriented, ranking friendly, localization, result ranking optimization, outsourcing and human powered in addition to locally resource utilization capacity to mouth more personalized, graphical and cached driven information from the web. This crawler is destining to present a framework, which will convince the chaffing and marketing experience while rendering the Web [6, 7,11].

2. Related work

A search engine robot's action is called spidering, as it resembles the multiple legged spiders. The spider's job is to go to a web page, read the contents and other info like user behavior, connect to any other pages on that web site

through links, and bring back the information. From one page it will travel to several pages and this proliferation follows several parallel and nested paths simultaneously.

Although release about the functioning issues of professional search engines is merely available and if available then it is only up to the concept level [6,11]. 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].

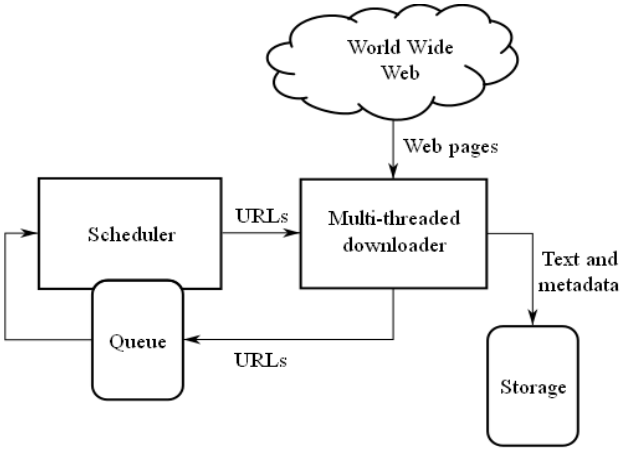

Fig.1 Components of standard Information Retrieval System [6, 10]

Storage: 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.

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

- To keep track of the URLs that has to be crawled.
- To actually fetch the content of the URL to be crawled
- To parse the retrieved URL.

Downloader: This component is responsible to download the corresponding URL as per priority in the queue scheduled by scheduler component and then to fetch metadata from considered html page. Output of this component becomes an input to storage module.

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. Extended Framework

Building an effective web crawler to solve your purpose is not a difficult task, but choosing the right strategies and building an effective architecture will lead to implementation of highly featured web crawler application [10]. The minimal scheme outlined above for crawling demands several modules that fit together are (see Fig.2).

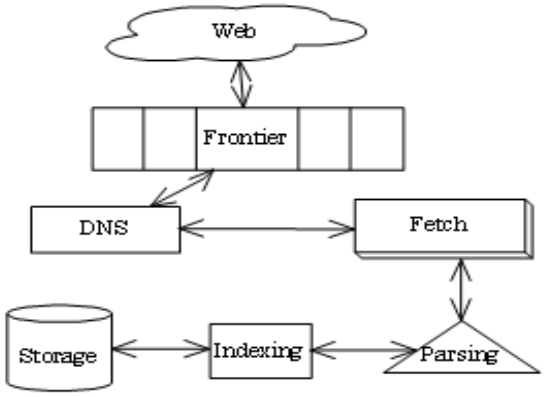

Fig.2 Architecture of different modules of Crawler [11]

1. The URL frontier, containing URLs yet to be fetched in the current crawl (in the case of continuous crawling, a URL may have been fetched previously but is back in the frontier for re-fetching).

2. A DNS resolution module that determines the web server from which to fetch the page specified by a URL

3. A fetch module that uses the http protocol to retrieve the web page at a URL.

4. A parsing module that extracts the text and set of links from a fetched web page.

5. A duplicate elimination module (Indexing module) that determines whether an extracted link is already in the URL frontier or has recently been fetched.

3.1 Extended Framework with URL Tracking

Url tracking is the process to implement the indexed database created by index agent of the crawler as a vertex while starting from seed url and finally a graph having internal links as an in-degree and external links as an out-degree of the current vertex. This is a logical mapping between meta info of web pages and their graph representation weighted with values of ToUrl, FromUrl, ContextKeyword and sever location. This is a module embedded with searching module of a crawler to be started with initial Url (see Fig.3). Pseudocode for this module is described below:

```
Ask user or automation module to specify the seed URL that crawler should crawl.
Add the URL to the empty list of URLs to search.
While not empty (the list of URLs to search)
{       Take the first URL in from the list of URLs.
        If the URL protocol is not HTTP then
                break;
                go back to while;
        If robots.txt file exist on site then
                If file includes .Disallow. statement then
                break;
                go back to while;
        Open the URL;
        If the opened URL is not HTML file and not explicitly requested file then
                Break;
                Go back to while;
        Iterate the HTML file;
        While the html text contains another link
        {
        If robots.txt file exist on URL/site then
        If file includes .Disallow. statement then
                break;
                go back to while;
If the opened URL is HTML file or explicitly requested file then
If the URL isn't marked as searched then
Mark this URL as already searched URL.
Insert new record to the table while maintaining internal links, external links and context weight associated with keyword.
                Else
Update existing record in the list.
        }
 }
```


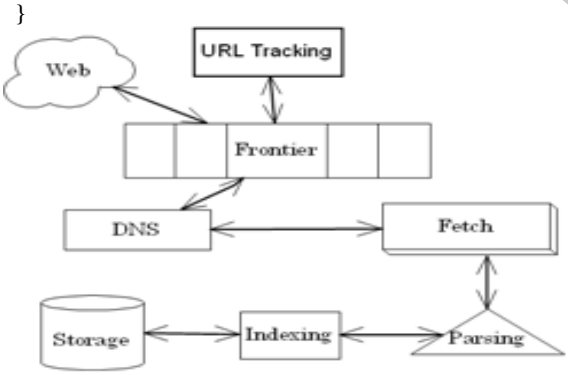

Fig.3 Crawler with URL Tracking Module

3.1.1 User Presentation Module

User Presentation Module is the subprogram that is responsible to fetch and represent the result on the bases of keywords directed by user's query from indexed database of search engine (see Fig.4). Pseudocode for this module is described below:

```
Ask user presentation module to specify the query.
Split query in the sets of keywords excluding stop words.
Add these sets to the empty list of keywords to search.
While not empty (the list of keywords to search)
{   Take the one set from the list and search in indexed database.
        If there is record matching to set then
            Fetch the record in output buffer.
}
If output buffer is NOT empty then
  Assemble result page with priority assigned on weights and display.
Else
No Result page is displayed.
```

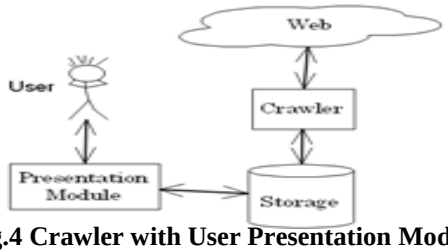

Fig.4 Crawler with User Presentation Module

3.2 Framework with URL Canonicalization

The URL canonicalization is a process that transforms a URL into a canonical form that is normal form (see Fig.5). Where URL is a composed of five components: the scheme, authority, path, query, and fragment components (see Fig.6). During the URL normalization, syntactically different URLs that are equivalent should be transformed into a syntactically identical URL (see Fig.7)[10,21].

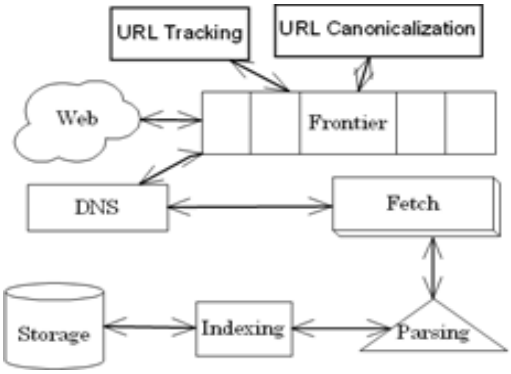

Fig.5 Crawler with URL Canonicalization Module

Modified pseudocode for tracking module with this module is described below:

```
Ask user or automation module to specify the seed URL that crawler should crawl.
Perform all possible normalizations to the seed URL
Add the normalized URL to the empty list of URLs to search.
While not empty (the list of URLs to search)
{       Take the first URL in from the list of URLs.
            If the URL protocol is not HTTP then
                        break;
                        go back to while;
            If robots.txt file exist on site then
                        If file includes .Disallow. statement then
                        break;
                        go back to while;
            Open the URL;
            If the opened URL is not HTML file and not explicitly requested file then
                        Break;
                        Go back to while;
            Iterate the HTML file;
            While the html text contains another link
{
                        If robots.txt file exist on URL/site then
                        If file includes .Disallow. statement then
                        break;
                        go back to while;
                        If the opened URL is HTML file or explicitly requested file then
                        If the URL isn't marked as searched then
                        Mark this URL as already searched URL.
Insert new record to the table while maintaining internal links, external links and context weight associated with keyword.
                        Else
Update existing record in the list.
            }
  }
```

On the basis of the identical meaning of the URL components, normalization can be categorized as follows:

URL example	http://mypunia.com:8065/apnasearch/result?rid=r102#d1				
URL components	Scheme	Authority	Path	Query	Fragment
URL Components example	http	mypunia.com:8065	apnasearch/result	rid=r102	d1

Fig.6 URL Components

Normalization	URL example before normalization	Identical URL after normalization
Path String	http://mypunia.com	http://mypunia.com/
Fragment	http://mypunia.com/search.php#d1	http://mypunia.com/search.php
Default Port	http://mypunia.com:80	http://mypunia.com
Case	HTTP://Mypunia.com	http://mypunia.com
Percent Encoding	http://mypunia.com/%7Eroot	http://mypunia.com/~root
Path Segment	http://mypunia.com/p1//p3.php	http://mypunia.com/p1/p3.php
Deemed Value	http://mypunia.com/p1/p2/./../p3.php	http://mypunia.com/p1/p3.php

Fig.7 URL Canonicalization

4. 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 (see Fig. 8) [6, 7,11].The abstraction (architecture of fetching module and presentation logic)of the developed architecture with extended features is detailed as stated further (see Fig. 9, Fig. 10).

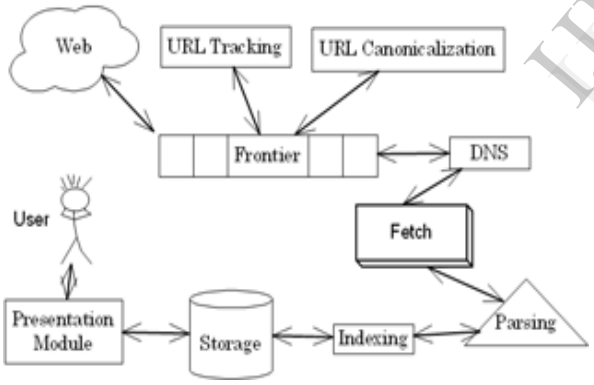

Fig.8 Framework with extended features

4.1 Architecture of Fetching Module

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 (see Fig. 9) [6,7].

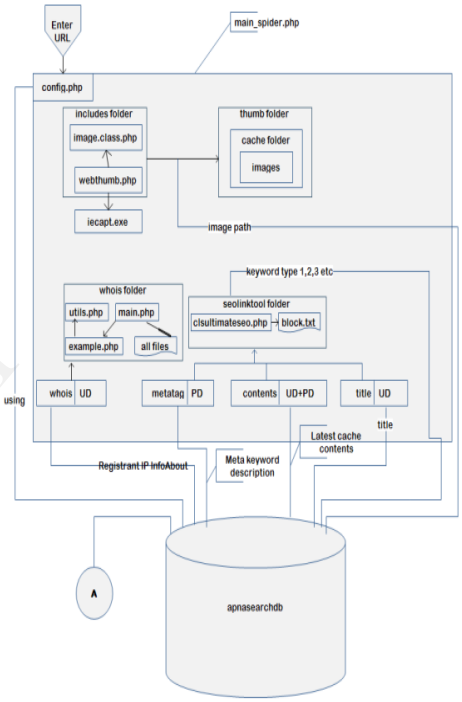

Fig. 9 Architecture of Fetching Module[16]

4.2 Presentation Logic Architecture

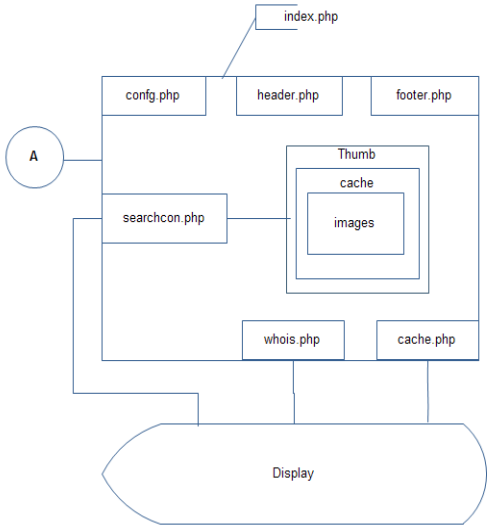

Fig. 10 Architecture of Interacting Agent

Presentation Logic: An interacting agent that gets keyword(s) to search indexed database and expel result page (see Fig.10)[11].

5. Performance

An estimated and approximate performance analysis can be done to compare the existing search strategies with the developed one. With the increase in 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. 11). The number of web pages given as the result of a user initiated will definitely grow up to an extent.

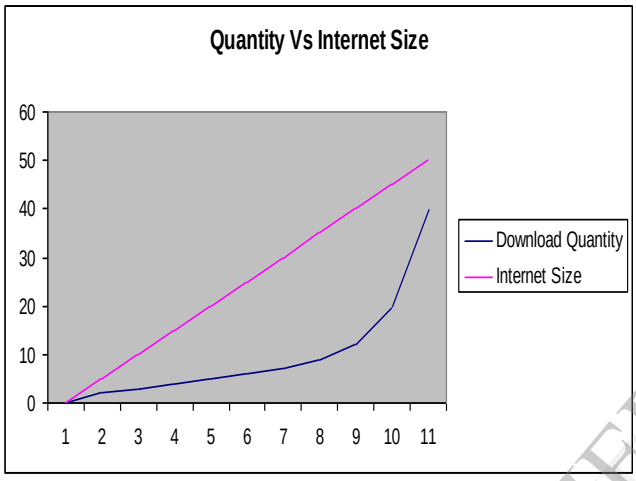

Fig. 11 Download Quantity vs. Internet Size.

This increase in the quantity on one hand, leads to decrease in the quality (see Fig. 12) 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.

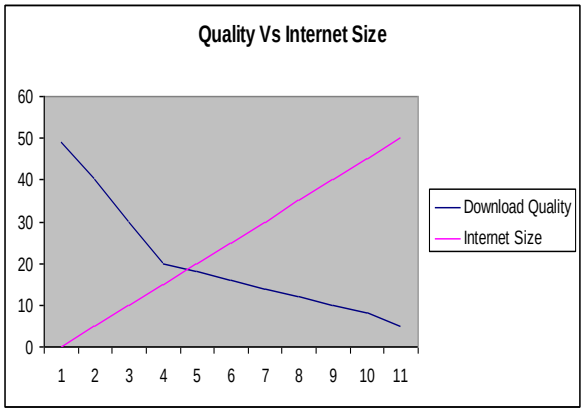

Fig. 12 Download Quality vs. Internet Size

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, path-oriented, cross architecture, cross platform and graphical.

5.1 Experimental Screenshots

A series of user interfaces of developed framework with deployed Extended CurlCrawler(see Fig. 13, Fig. 14, Fig. 15, Fig. 16) while rendering for a keyword "song" is shown below:

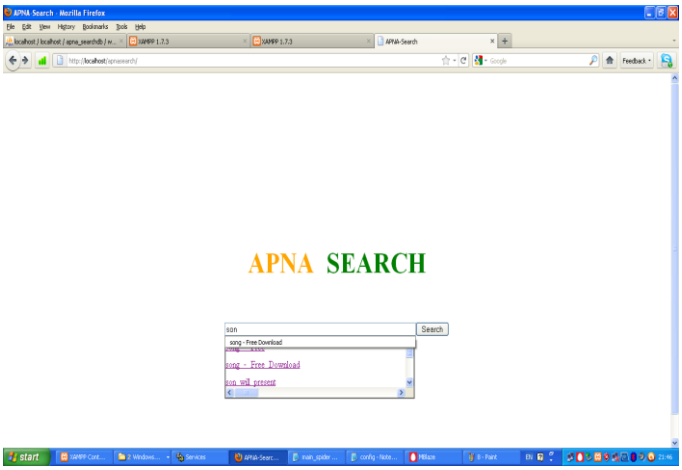

Fig. 13 Home Interface

Fig. 14 Thumb Created Result

Fig. 15 WhoIs Info Result

Fig. 16 Cache Result

5.2 Analysis

This optimized 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,IIS 7.5,Tomcat 7.0.23,Asp.Net run time framework 4.0,SQL Server 2008 and XAMPP 1.7.3.

In this paper, experimental statistics are presented of 9 days only owing to compare with other existing search systems like Google and previous developed framework called CurlCrawler, about these requests 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] and CurlCrawler made 67.3 millions HTTP requests in 9 days, achieving an average download rate of 87.52 docs/sec and 376.45 KB/sec. Performance of any information retrieval system can be analyzed using parameters like coverage, user perception that are presented below:

5.2.1 Coverage

Coverage of a search engine points towards a search engine's crawl speed and index size. In case of developed framework, optimized CurlCrawler made 70.6 millions HTTP requests in 9 days, achieving an average download rate of 89.97 docs/sec and 396.35 KB/sec. Hence, this work with local resource utilization features of URL tracking and normalization is a forward step in the process of considerable optimization mark and represented as below (see Fig. 17):

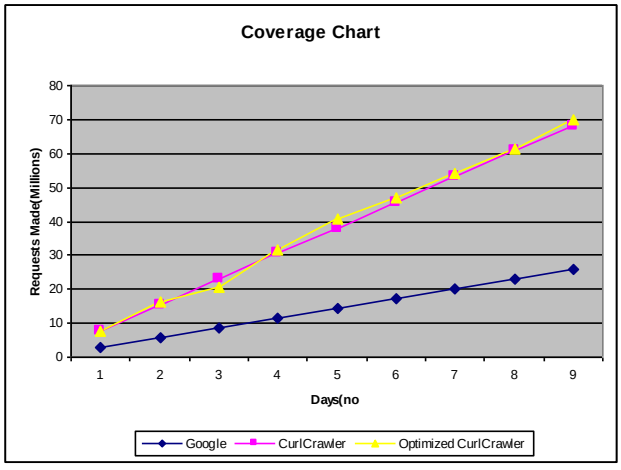

Fig. 17 Coverage Chart

5.2.2 User Perception

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

GUI perception

Out of 70.6 million requests made, 2.51 millions requests do not return thumb i.e.3.5552% and 1.47 millions requests return a thumb that is not clear up to the identifying mark i.e. 2.0821%(see Fig. 18).

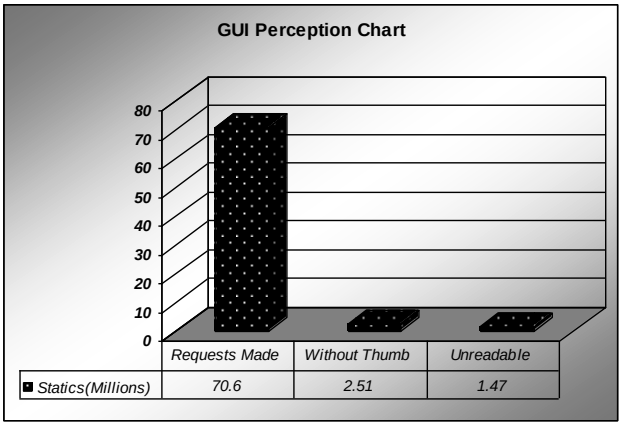

Fig. 18 GUI Perception Chart

Personalization degree

Out of 70.6 million requests made, 2.35 millions requests do not return personalization of information like registrant, hosting info etc i.e. 3.3286 %(see Fig. 19).

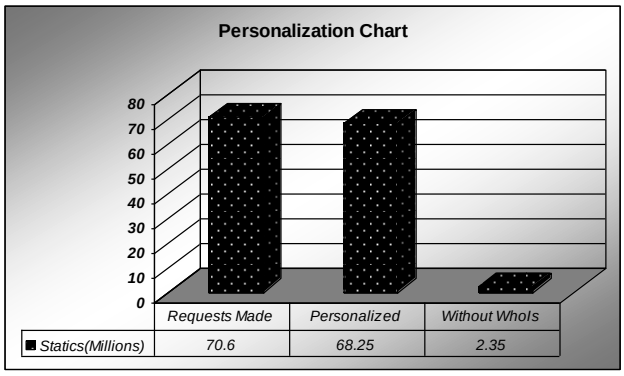

Fig. 19 Personalization Chart

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

6. Conclusion

In addition to the information like thumb, cache, registrant and higher degree of context to provide more interesting perception from users interacting with, this optimized framework renders the web with forward-backward tracking and URL normalization oriented approach to provide a cross architecture framework for search engines. This is a part of ongoing research work, to utilize advance features of programming in the web crawling up to maximum extent of efficiency. 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 fetching module.

6.1 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
D1	Div	To store corresponding keyword from database to implement AJAX while rendering
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;
```

6.2 Fetch

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	24

Pseudo code:

```
read url;
urlNormalization(url);
if(validateApproach(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']['do
main']['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 optimized framework along with implementation details of various modules used is discussed. An optimized crawler executing in a Multi-Agent environment is designed and developed to expel a search that is more focused, path-oriented, relevant, personalized, cached and GUI driven. An extension to the developed framework is also going on that uses an additional agent named URL mining and ontology with features of reference or knowledge discovery, 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 [6,11].

7. References

[1].Heydon, A. and Najork, M., 1999. Mercator: A Scalable, Extensible Web Crawler, International Journal of WWW, Vol. 2. No. 4. (1999) 219-229

[2].Berners-Lee, T., Fielding, R., and Masinter, L.: Uniform Resource Identifiers(URI):GenericSyntax , (2005)

[3].Kim, S.J. and Lee, S.H.: An Empirical Study on the Change of Web Pages, Springer-Verlag Lecture Notes in Computer Science, Vol. 3399. (2005) 632-642

[4].Lee, S.H., Kim, S.J. and Hong, S.: On URL Normalization, Springer-Verlag Lecture Notes in Computer Science, Vol. 3481. (2005) 1076-1085

[5].Burner, M.: Crawling Towards Eternity: Building an Archive of the World Wide Web, Web Techniques Magazine, Vol. 2. No. 5. (1997) 37-40

[6].Ela Kumar, Ashok Kumar al,(IJCSIT)International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 17001705(ISSN: 0975-9646).

[7].Ela Kumar, Ashok Kumar al, (IJCTT) International Journal of Computer Trends and Technology- Vol. 3(3)-2012,342-350 (ISSN: 2231-2803).

[8].M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In SIGI '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 284{291, August 2006.

[9].G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 141{150, May 2007.

[10].Ashok Kumar al, (IJSECT)International Journal of Science, Engineering and Computer Technology- Vol. 2(3)-2012,33-38 (ISSN: 2229-4937).

[11].Ashok Kumar al, (IJSECT)International Journal of Science, Engineering and Computer Technology- Vol. 2(3)-2012,43-47 (ISSN: 2229-4937).

[12].Dr. Ela Kumar et al. /International Journal of Engineering and Technology Vol.2 (2), 2010, 38-44.

[13].Kim, S.J. and Lee, S.H.: Implementation of a Web Robot and Statistics on the Korean Web, Springer-Verlag Lecture Notes in Computer Science, Vol.2713. (2003) 341-350

[14] A. C. Carvalho, E. S. Moura, A. S. Silva, K. Berlt, And A. Bezerra. A cost-effective method for detecting web site replicas on search engine databases. Data Knowl. Eng.,62(3):421–437, 2007.

[15] A. Chowdhury, O. Frieder, D. A. Grossman, and M. C.McCabe. Collection statistics for fast duplicate document detection. TOIS, 20(2):171–191, 2002.

[16] A. Dasgupta, R. Kumar, and A. Sasturkar. De-duping URLs via rewrite rules. In KDD, pages 186–194,2008.

[17] M. Najork. Systems and methods for inferring Uniform resource locator (URL) normalization rules.US Patent Application Publication, 2006/0218143,Microsoft Corporation, 2006.

[18].Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the dust: di®erent urls with similar text. In WWW '07: Proceedings of the 16t international conference on World Wide Web, pages 111{120, May 2007.

[19].Shkapenyuk, V. and Suel, T.: Design and Implementation of a High-performance Distributed Web Crawler, In Proceedings of 18th Data Engineering Conference, (2002) 357-368

[20] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the dust: different URLs with similar text.In WWW, pages 111–120, 2007.

[21] Sung Jin Kim, Hyo Sook Jeong, Sang Ho Lee. Reliable Evaluations of URL.Journal of Security Engineering,Vol. 2, No. 1, November, 2005.

[22] A. Broder, S. C. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. Computer Networks, 29(8–13):1157–1166, 1997.