Extended CurlCrawler: A focused and path-oriented framework for crawling the web with thumb.

Dr Ela Kumar[#], Ashok Kumar[$]
# School of Information and Communication Technology
Gautam Buddha University, Greater Noida(India)
$ AIM & ACT.
Banasthali University,Banasthali(Raj.)-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 exposed and up-to-date huge repository of information available to everyone, everywhere and every time [1]. It is the thrust arena of engineering endeavor and is evolving without a grand design blueprint. Finally, an age has come, where information has become an instrument, a tool that can be used to solve many problems. The biggest challenge being posed by the Internet is its ever-growing size with the availability of endless pool of information hosted on the World Wide Web (WWW). 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 reduced chaffing and cross features of the framework. With further increase in the size of the Internet, the problem grows exponentially. Crawlers can retrieve data much quicker and in greater depth than human searchers, so they can have a crippling impact on the performance of a site [7, 17]. Needless to say that 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 multi-agent framework to outcome highly featured web crawler application [2, 3].**

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, focused, path oriented and human powered.

*Keywords and Phrases-***Topical, SOAP, Interacting Agent, WSDL, Thumb, Whois, CachedDatabase, IECapture, Searchcon, Main_spider, UDDI.**

1. Introduction

WWW is immense to obtain information and moreover information on web is voyaged using search engines like AltaVista, WebCrawler, Hot Boat etc[1]. Owing to the reason that search engines are the striking one to sail the web for several purposes. Optimization of search engine is a raptorial field to address a state of fast growing rate of amount of information on the web. 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.

A typical web crawler starts by parsing a specified web page and noting any hypertext links on that page that point to other web pages. The Crawler then parses those pages for new links, and so on, recursively. A crawler is a software or script or automated program which resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. 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 rate of change on web pages are two important characteristics of the Web that generate a scenario in which Web crawling is very difficult. A large volume of web page implies that web crawler can only download a fraction of the web pages and hence it is very essential that web crawler should be intelligent enough to prioritize download.

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. The difficulties in implementing efficient web crawler clearly state that bandwidth for conducting crawls is neither infinite nor free. So, it is becoming essential to crawl the web in not only a scalable, but in an efficient way, if some reasonable amount of quality or freshness of web pages is to be maintained. This ensues that a crawler must carefully choose at each step which pages to visit next.

The aim of this paper is to raffle an extended 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 extended design and implementation of widened Curl Crawler, featured with cross platform, cross architecture, focused, path oriented 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 experience while searching on the Internet [16].

2. 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 [12]. The minimal scheme outlined above for crawling demands several modules that fit together are (see Fig.1).

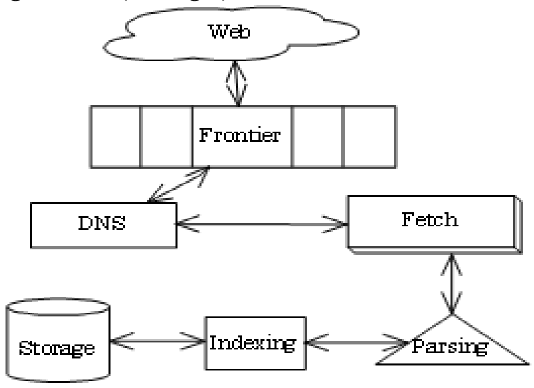


Fig.1 Architecture of different modules of Crawler [15]

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.

2.1 Automated with human powered approach

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, 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. Spiders frequent the site at some interval, may be a month to a few months, and re-index the pages. This way any changes that may have occurred in your pages could also be reflected in the index. The spiders automatically visit your web pages and create their listings. The spider's movement across web pages stores those pages in its memory, but the key action is in indexing. The index is a huge database containing all the information brought back by the spider. The index is constantly being updated as the spider collects more information. This automated task of spider may be done explicitly by human power and it may be considered as a remarkable for the situation where crawler program deficit to fetch owing to spam or stop words. Such types of systems are also categorized as directories. Framework developed for CurlCrawler employs automation with human powered approach to take advantages of both the approach. In plain words, implemented framework has two sets of listings based on both the mechanisms mentioned above (see Fig.2).

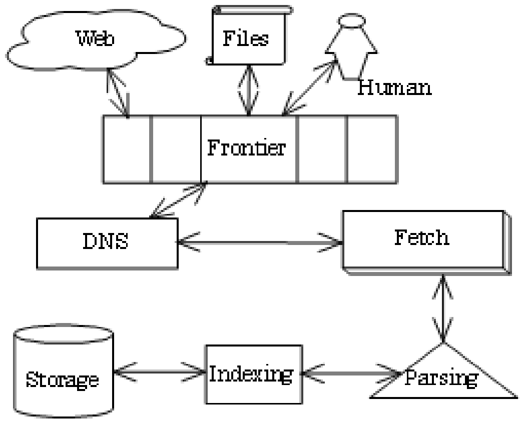


Fig.2 Extended framework with automated and human powered approach

2.2 Focused and Path Oriented

Path-ascending crawling intends the crawler to download as many resources as possible from a particular Web site. That way a crawler would ascend to every path in each URL that it intends to crawl. For example, when given a seed URL of http://abc.org/a/b/index.html, it will attempt to crawl /a/b/, /a/, and /. The advantage with Path-ascending crawler is that they are very effective in finding isolated resources, or resources for which no inbound link would have been found in regular crawling. The importance of a page for a crawler can also be expressed as a function of the similarity of a page to a given query. In this strategy we can intend web crawler to download pages that are similar to each other, thus it will be called focused crawler or topical crawler.

In order to take advantages of both the approach, framework developed for CurlCrawler employs two sets of listings based on both the approaches mentioned above. In plain words, implemented framework .Needless to say if a single crawler is performing multiple requests per second and/or downloading large files, a server would have a hard time keeping up with requests from multiple crawlers. To resolve this problem we are using robots exclusion protocol, also known as the robots.txt protocol [13].

2.3 Pseudo code for proposed web crawler

Here's a pseudo code summary of the algorithm that is used to implement proposed web crawler:

```
Ask user or automation module to specify the starting URL on web and file type 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 list.
                Else
Update existing record in the list.
        }
  }
```

2.4 Outsourced

To utilize the component oriented features of this eras programming and expose it with cross architecture and cross platform features this work is deployed using web service (see Fig.4). Web Services are a general model for building applications and can be implemented for any operation system that supports communication over the Internet [12,14]. Web services take advantage of the best of component-based development. Component-based object models like Distributed Component Object Model (DCOM), Remote Method Invocation (RMI), and CORBA's Internet Inter-ORB Protocol (IIOP) have been around for some time. Unfortunately, they depend on an object model–specific protocol [13]. Web services extend these models by communicating with Simple Object Access Protocol (SOAP) and XML to eradicate the barrier posed by the object model–specific protocol.

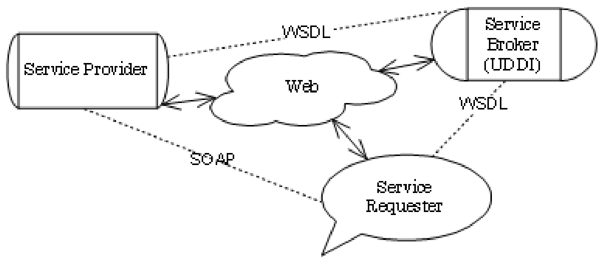


Fig.3 Web service architecture.

Web services work by basically using HTTP and SOAP to make business data available on the Web. Web services expose business objects (COM objects, JavaBeans, etc.) to SOAP calls over HTTP and execute remote function calls (see Fig.3). That way, Web service consumers are able to invoke method calls on remote objects by using SOAP and HTTP over the Web.

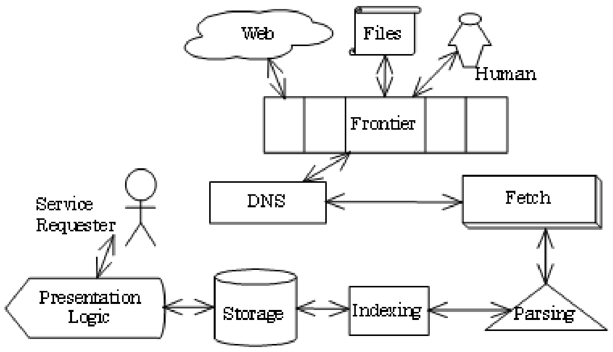


Fig.4 Framework with outsourced approach

3. Architecture of extended CurlCrawler

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. 5) [6, 8, 16].The abstraction of the developed architecture with extended features is detailed as stated further (see Fig. 6, Fig. 7, Fig. 8) with modules having keynote priority.

3.1 Architecture of Fetching Module.
3.2 Cached Database Architecture.
3.3 Presentation Logic Architecture.

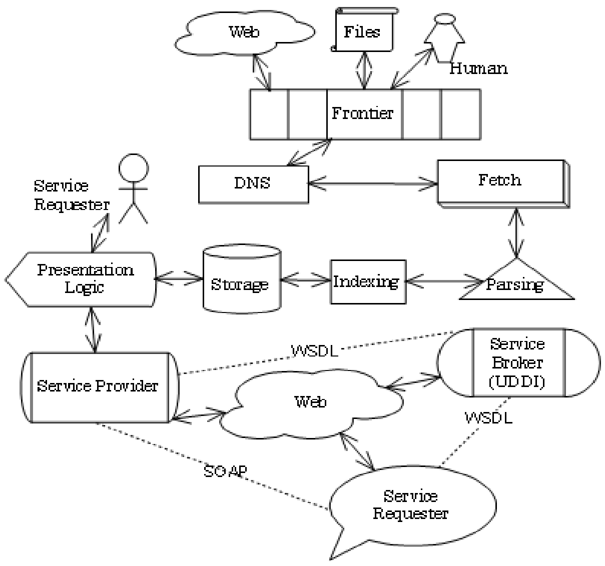


Fig.5 Framework with extended features

3.1 Architecture of Fetching Module

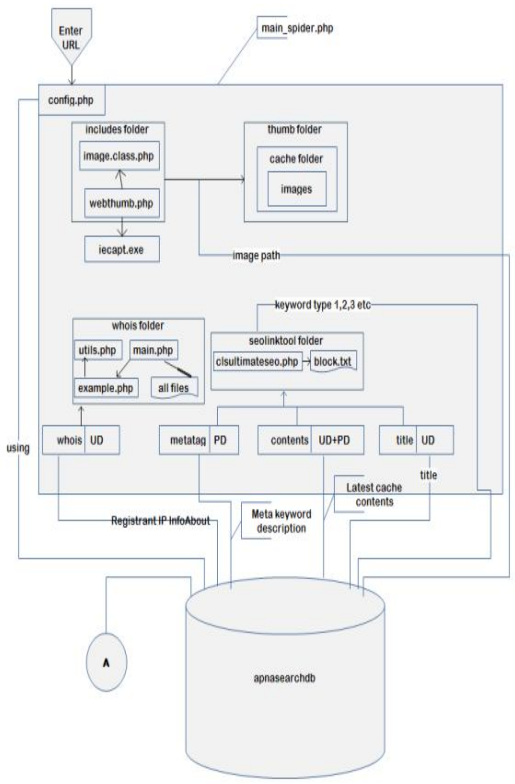


Fig. 6 Architecture of Fetching Module[16]

Fetch: 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. 6) [5,7].

3.2 Cached database architecture

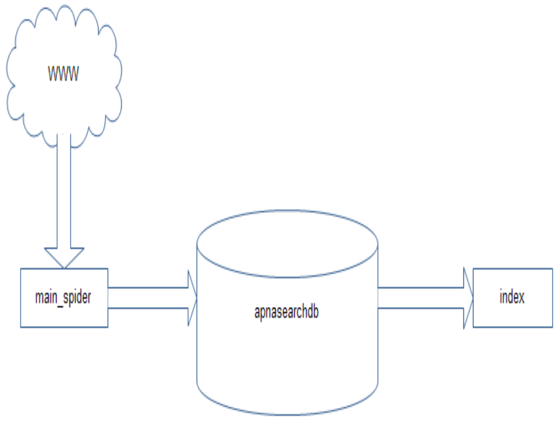


Fig.7 Architecture of Cached Database[16]

Cached Storage: An agent that employs a module named as Index, a filtering module that provides user perception and interest to be used to fetch result from database server (see Fig.7).

3.3 Presentation Logic Architecture

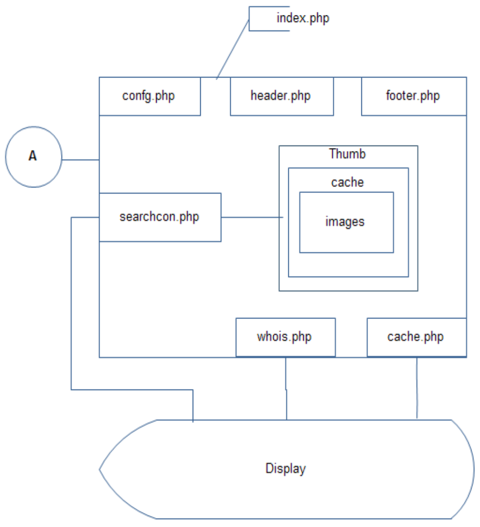


Fig. 8 Architecture of Interacting Agent

Presentation Logic: An interacting agent that gets keyword(s) to search indexed database and expel result page (see Fig.8)[16].

4. 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. 9). The number of web pages given as the result of a user initiated will definitely grow up to an extent.

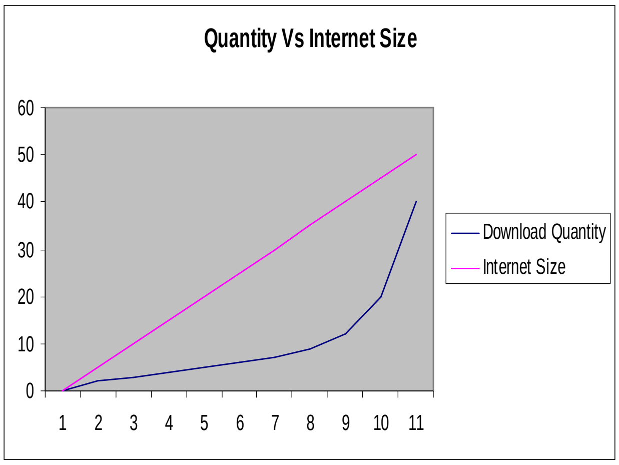


Fig. 9 Download Quantity vs. Internet Size.

This increase in the quantity on one hand, leads to decrease in the quality (see Fig. 10) 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.

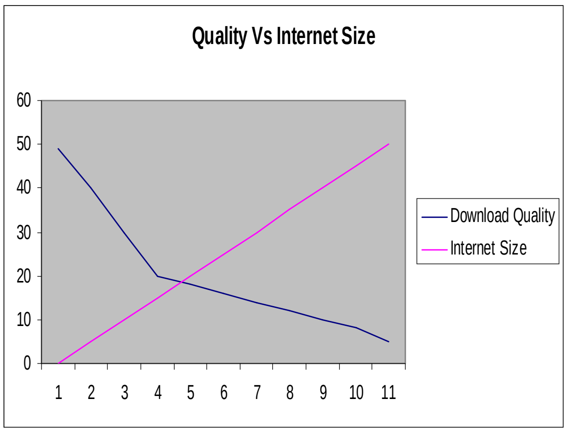


Fig. 10 Download Quality vs. Internet Size

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.

4.1 Experimental Screenshots

A series of user interfaces of developed framework with deployed Extended Curl Crawler(see Fig. 11, Fig. 12, Fig. 13, Fig. 14) while rendering for a keyword "song" is shown below:

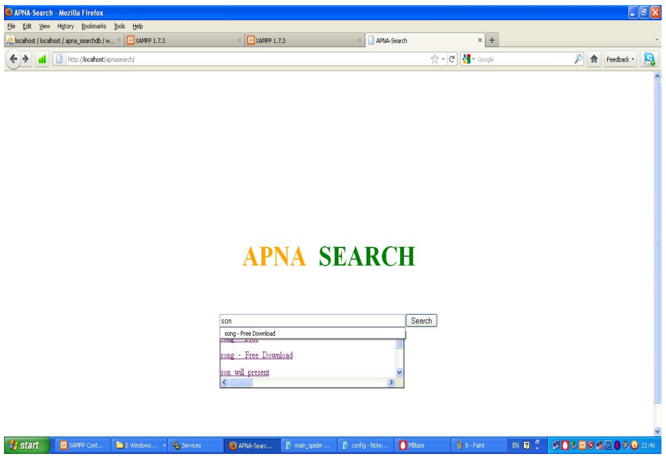
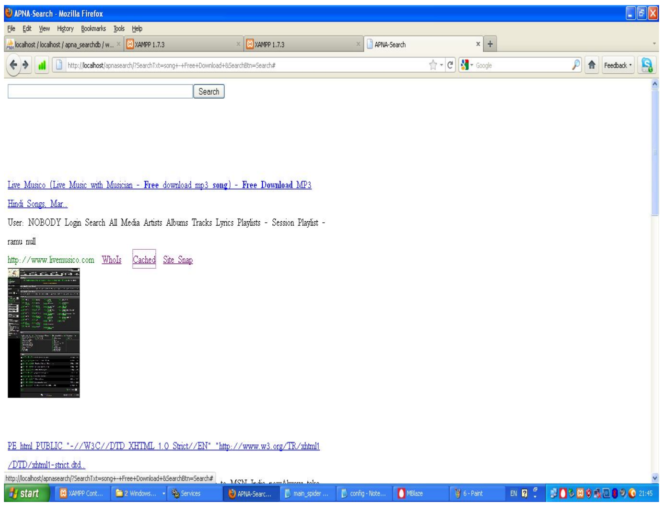


Fig. 11 Home Interface


APNA-Search - Mozilla Firefox
http://localhost/apnasearch/?SearchTxt=song+-+Free+Download+&SearchBtn=Search#
Search
start
XAMPP Cont...
2 Windows...
Services
APNA-Searc...
main_spider ...
config - Note...

Live Musico (Live Music with Musician - **Free** download mp3 **song**) - **Free Download** MP3 Hindi Songs, Mar..

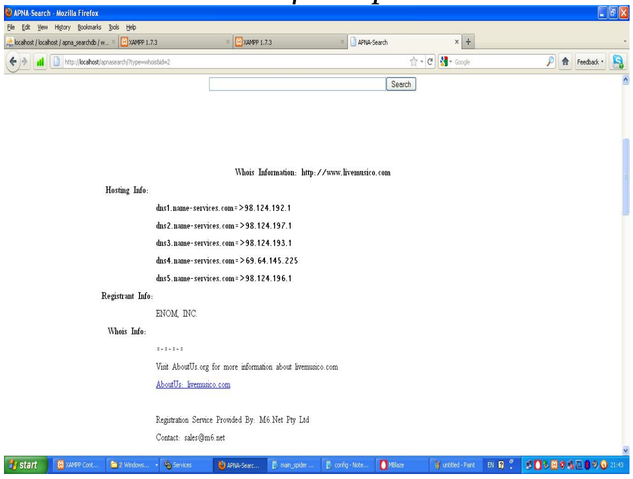
User: NOBODY Login Search All Media Artists Albums Tracks Lyrics Playlists - Session Playlist - ramu null

http://www.livemusico.com WhoIs Cached Site Snap

PE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd..

http://localhost/apnasearch/?SearchTxt=song+-+Free+Download+&SearchBtn=Search#

Fig. 12 Thumb Created Result


APNA-Search - Mozilla Firefox
http://localhost/apnasearch/?type=whois&id=2
Search

Whois Information: http://www.livemusico.com

Hosting Info:

dns1.name-services.com=>98.124.192.1

dns2.name-services.com=>98.124.197.1

dns3.name-services.com=>98.124.193.1

dns4.name-services.com=>69.64.145.225

dns5.name-services.com=>98.124.196.1

Registrant Info:

ENOM, INC.

Whois Info:

=-=-=-=

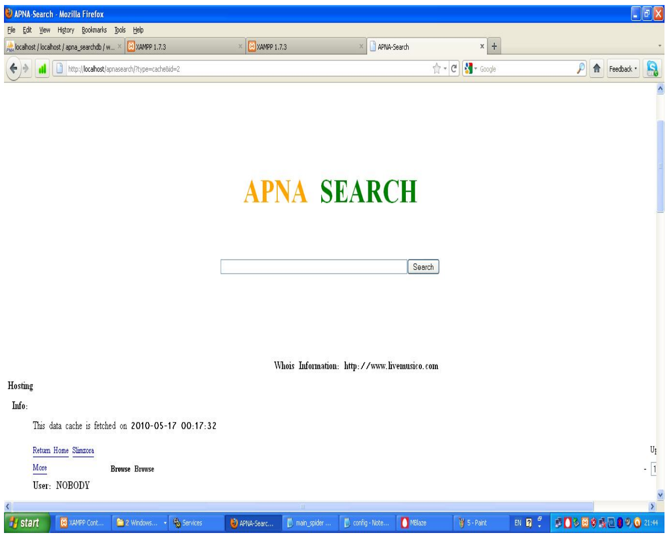
Visit AboutUs.org for more information about livemusico.com

AboutUs: livemusico.com

Registration Service Provided By: M6.Net Pty Ltd

Contact: sales@m6.net

Fig. 13 WhoIs Info Result


APNA-Search - Mozilla Firefox
http://localhost/apnasearch/?type=cache&id=2
APNA SEARCH
Search

Whois Information: http://www.livemusico.com

Hosting

Info:

This data cache is fetched on 2010-05-17 00:17:32

Return Home Slimzora

More Browse Browse

User: NOBODY

Fig. 14 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,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 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, Extended CurlCrawler made 67.3 millions HTTP requests in 9 days, achieving an average download rate of 87.52docs/sec and 376.45 KB/sec includes explicit mining option with focused and path-oriented approach. Hence, this work with local resource utilization is a considerable optimization mark and represented as below (see Fig. 15):

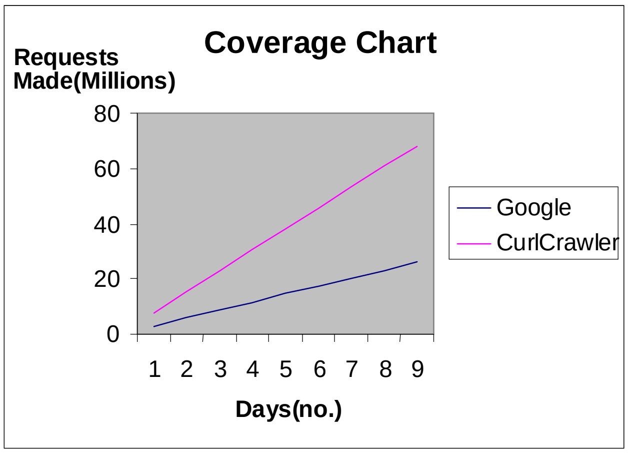


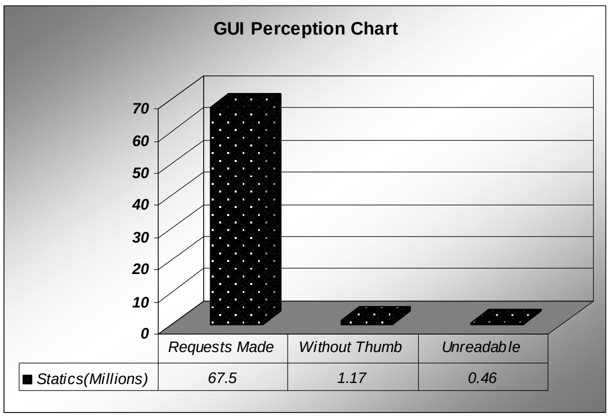
Fig. 15 Coverage Chart

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 67.5 million requests made, 1.17 millions requests do not return thumb i.e.0.785% and 0.46 millions requests return a thumb that is not clear up to the identifying mark i.e. 0.31%(see Fig. 16).



	Requests Made	Without Thumb	Unreadable
■ Statics(Millions)	67.5	1.17	0.46

Fig. 16 GUI Perception Chart

Personalization degree

Out of 67.5 million requests made, 0.07 millions requests do not return personalization of information like registrant, hosting info etc i.e.0.047 %(see Fig. 17).

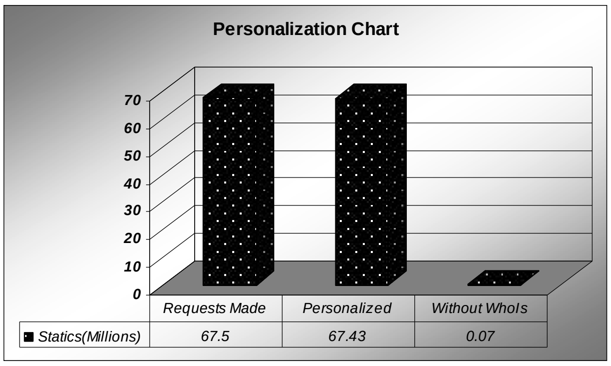


Fig. 17 Personalization Chart

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

5. 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 extended framework renders the web with focused and path oriented approach to provide a cross architecture framework for search engines powered with human opinion approach. . 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.

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
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;
```

5.2 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	22

Pseudo code:

```
read 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 extended framework along with implementation details of various agents used is discussed. An extended crawler executing in a Multi-Agent environment is designed and developed to expel a search that is more focused, path-oriented, relevant, personalized, cached, automated, opinion mined with human power, cross architecture, cross platform and GUI driven. An extension to the developed framework is also going on that uses an additional agent named Learner Agent with features of Artificial Intelligence, 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]. Segev, Elad (2010). Google and the Digital Divide: The Biases of Online Knowledge, Oxford: Chandos Publishing.

[2].Vaughan, L. & Thelwall, M. (2004). Search engine Coverage bias: evidence and possible causes, Information Processing &Management,40(4), 693-707.

[3].Gandal, Neil (2001). "The dynamics of competition in the internet search engine market". *International Journal of Industrial Organization* 19 (7): 1103–1117.

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

[5].Steve Lawrence; C. Lee Giles (1999). "Accessibility of information on the web". Nature 400 (6740): 107–9.

[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].Cho,Junghoo,"Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data", Ph.D. dissertation, Department of Computer Science, Stanford University, November 2001.

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

[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].Gray, N. A. B. (2005). "Performance of Java Middleware - Java RMI, JAXRPC, and CORBA". University of Wollongong. pp. 31–39. Retrieved January 11, 2011.

[13].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.

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

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

[16].Ela Kumaret al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1700-1705.

[17].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.

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