wjert, 2017, Vol. 3, Issue 6, 345-356. *Review Article* **ISSN 2454-695X**

World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 4.326

DESIGN AND IMPLEMENTATION OF A HIGH PERFORMANCE WEB CRAWLER FOR INFORMATION EXTRACTION

[1]*ILO Somtoochukwu F., [2]Victor Onuchi, [3]Akuma Uche and [4]Okah, Paul-Kingsley

[1,2,3]Computer Engineering Department, Michael Okpara University of Agriculture, Umudike, Abia State, Nigeria.

[4]Electronic and Computer Engineering Department, Nnamdi Azikiwe University Awka, Anambra State, Nigeria.

Article Received on 24/09/2017 **Article Revised on 17/10/2017** **Article Accepted on 07/11/2017**

***Corresponding Author**
ILO Somtoochukwu F.
Computer Engineering Department, Michael Okpara University of Agriculture, Umudike.

ABSTRACT

Broad web search engines as well as many more specialized search tools rely on web crawlers to acquire large collections of pages for indexing and analysis. Such a web crawler may interact with millions of hosts over a period of weeks or months, and thus issues of robustness, flexibility, and manageability are of major importance. In addition, I/O performance, network resources, and operating system limits must be taken into account in order to achieve high performance at a reasonable cost. In this study, we describe the design and implementation of a high performance web crawler that runs on a network of workstations. The crawler scales to (at least) several hundred pages per second, is resilient against system crashes andother events, and can be adapted to various crawling applications. We present the software architecture of the system, discuss the performance bottlenecks, and describe efficient techniques for achieving high performance. An algorithm was developed for the web crawler to download web pages that are to be indexed by the search engine and based on the algorithm developed above, the source code was written in the PHP scripting language that is suited for web development and can be embedded into HTML. The source code was then integrated in Apache Server Environment for automatic web search engine fetch sequencing. The program workability was test run in OSI model layer 7 using Hypertext Transfer Protocol (HTTP).

KEYWORDS: Web Crawler, Web Search Engine, PHP Scripting Language, Apache Server Enviroment.

I. INTRODUCTION

A **Web crawler**, sometimes called a **spider**, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing (web spidering). Web search engines and some other sites use Web crawling or spidering software to update their web content or indices of others sites' web content. Web crawlers can copy all the pages they visit for later processing by a search engine which indexes the downloaded pages so the users can search much more efficiently.

Crawlers consume resources on the systems they visit and often visit sites without approval. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For instance, including a robots.txt file can request bots to index only parts of a website, or nothing at all.

As the number of pages on the internet is extremely large, even the largest crawlers fall short of making a complete index. For that reason search engines were bad at giving relevant search results in the early years of the World Wide Web, before the year 2000. This is improved greatly by modern search engines, nowadays very good results are given instantly.Web crawlers are an important component of web search engines, where they are used to collect the corpus of web pages indexed by the search engine. Moreover, they are used in many other applications that process large numbers of web pages, such as web data mining, comparison shopping engines, and so on. Despite their conceptual simplicity, implementing high-performance web crawlers poses major engineering challenges due to the scale of the web. In order to crawl a substantial fraction of the “surface web” in a reasonable amount of time, web crawlers must download thousands of pages per second, and are typically distributed over tens or hundreds of computers. Their two main data structures – the “frontier” set of yet-to-be-crawled URLs and the set of discovered URLs – typically do not fit into main memory, so efficient disk-based representations need to be used. Finally, the need to be “polite” to content providers and not to overload any particular web server, and a desire to prioritize the crawl towards high-quality pages and to maintain corpus freshness impose additional engineering challenges.

Conceptually, the algorithm executed by a web crawler is extremely simple: select a URL from a set of candidates, download the associated web pages, extract the URLs (hyperlinks) contained therein, and add those URLs that have not been encountered before to the candidate set. Indeed, it is quite possible to implement a simple functioning web crawler in a few lines of a high-level scripting language such as Perl.

However, building a web-scale web crawler imposes major engineering challenges, all of which are ultimately related to scale. In order to maintain a search engine corpus of say, ten billion web pages, in a reasonable state of freshness, say with pages being refreshed every four weeks on average, the crawler must download over 4,000 pages/second. In order to achieve this, the crawler must be distributed over multiple computers, and each crawling machine must pursue multiple downloads in parallel. But if a distributed and highly parallel web crawler were to issue many concurrent requests to a single web server, it would in all likelihood overload and crash that web server. Therefore, web crawlers need to implement politeness policies that rate-limit the amount of traffic directed to any particular web server (possibly informed by that server's observed responsiveness). There are many possible politeness policies; one that is particularly easy to implement is to disallow concurrent requests to the same web server; a slightly more sophisticated policy would be to wait for time proportional to the last download time before contacting a given web server again.

In some web crawler designs, the page downloading processes are distributed, while the major data structures – the set of discovered URLs and the set of URLs that have to be downloaded – are maintained by a single machine. This design is conceptually simple, but it does not scale indefinitely; eventually the central data structures become a bottleneck. The alternative is to partition the major data structures over the crawling machines. Ideally, this should be done in such a way as to minimize communication between thecrawlers. One way to achieve this is to assign URLs to crawling machines based on their host name. Partitioning URLs by host name means that the crawl scheduling decisions entailed by the politeness policies can be made locally, without any communication with peer nodes. Moreover, since most hyperlinks refer to pages on the same web server, the majority of links extracted from downloaded web pages is tested against and added to local data structures, not communicated to peer crawlers.

In general, URLs should be crawled in such a way as to maximize the utility of the crawled corpus. Factors that influence the utility are the aggregate quality of the pages, the demand for certain pages and topics, and the freshness of the individual pages. All these factors should be considered when deciding on the crawl priority of a page: a high-quality, highly-demanded and fast-changing page (such as the front page of an online newspaper) should be recrawled frequently, while high-quality but slow-changing and fast-changing but low-quality pages should receive a lower priority.

II. LITERATURE REVIEW

RBSE which was the first published Web crawler was developed by Eichmann (1994). It was based on two programs: the first program, "spider" maintains a queue in a relational database, and the second program "mite", is a modified www ASCII browser that downloads the pages from the Web. The challenges encountered in absorbing 2.5 million (and eventually 4 million) lines of legacy FORTRAN have provided very useful data concerning scalability of these techniques and tools. The use of non-standard approaches to modularity in the existing FORTRAN has been major impediment to effective use of many of the tool suites.

Pinkerton (1994) developed WebCrawler which was used to build the first publicly-available full-text index of a sub-set of the Web. It was based on lib-WWW to download pages, and another program to parse and order URLs for breadth-first exploration of the Web graph. It also included a real-time crawler that followed links based on the similarity of the anchor text with the provided query.

World Wide Web Worm which was a crawler used to build a simple index of document titles and URLs. The index could be searched by using the grep UNIX command. It was Implemented by McBryan (1994).

Burner (1997) developed is a crawler designed with the purpose of archiving periodic snapshots of a large portion of the Web. It uses several process in a distributed fashion, and a fixed number of Web sites are assigned to each process. The inter-process exchange of URLs is carried in batch with a long time interval between exchanges, as this is a costly process. The Internet Archive Crawler also has to deal with the problem of changing DNS records, so it keeps an historical archive of the hostname to IP map pings.

WebSPHINX which was composed of a Java class library that implements multi-threaded Web page retrieval and HTML parsing, and a graphical user interface to set the starting

URLs, to extract the downloaded data and to implement a basic text-based search engine. It was developed by Miller and Bharat (1998).

Brin and Page (1998) created Google Crawler whose early version of its architecture was based in C++ and Python. The crawler was integrated with the indexing process, because text parsing was done for full-text indexing and also for URL extraction. There is an URL server that sends lists of URLs to be fetched by several crawling processes. During parsing, the URLs found were passed to a URL server that checked if the URL have been previously seen. If not, the URL was added to the queue of the URL server.

Da Silva et al (1999) developed CobWeb which uses a central "scheduler" and a series of distribute d "collectors". The collectors parse the downloaded Web pages and send the discovered URLs to the scheduler, which in turns assign them to the collectors. The scheduler enforces a breadth-first search order with a politeness policy to avoid overloading Web servers. The crawler is written in Perl.

In 2001WebFountain was developed which is a distributed, modular crawler similar to Merca tor but written in C++. It features a "controller" machine that coordinates a series of "ant" machines. After repeatedly downloading pages, a change r ate is inferred for each page and a non-linear programming method must be used to solve the equation system for maximizing freshness. The authors recommend to use this crawling order in the early stages of the crawl, and then switch to a uniform crawling order, in which all pages being visited with the same frequency.

III. MATERIALS AND METHODS

This new system is designed to aid the user because it is user friendly, all you do is follow the instruction, then click, it is interactive and menu driven. Having gathered and analyzed the required data explicitly, system specification, which determined what the system should be doing and how it should be design to achieved the targeted objectives is written down, the purpose of the design is to determine the specific requirement of the application identify and select the software and needed computer resources and completely and other necessary to develop the application. During the design phase, system requirement are determined in details.

The exact layout of report and document also selected and additional needed resources, such as assistance in developing one or more application or access to data in a software or a mainframe are identified. These and related requirements are specified during the design phase. The characteristics of the end user (i.e. input) computer application and how it is to functions should be carefully considered during the design phases in general, and user computing application should be designed with the following characteristics

- Easy to use and understand
- Visually attractive and easy to read
- Error assistant.
- Efficient in how to operate
- Completely documented with comments/ remarks
- Simple avoiding complicated logic and unnecessary loops.
- Compatible (to the extent possible) with other organization end user computing application or cooperate system
- Backed up to prevent the loss of important programs, data and work.

A. Software Algorithm

1. It checks for the next page to download – the system keeps track of pages to be downloaded in a queue.
2. Checks to see if the page is allowed to be downloaded- checking a robots exclusion file and also reading the header of the page to see if any exclusion instructions were provided do this. Some people don't want their pages archived by search engines.
3. Download the whole page.
4. Extract all links from the page (additional web site and page addresses) and add those to the queue mentioned above to be downloaded later.
5. Extract all words & save them to a database associated with this page, and save the order of the words so that people can search for phrases, not just keywords
6. Optionally filter for things like adult content, language type for the page, etc.
7. Save the summary of the page and update the last processed date for the page so that the system knows when it should re-check the page at a later stage.

B. Flow Chart

This phase of the project shows the procedure used to design the new system using charts, as shown below:

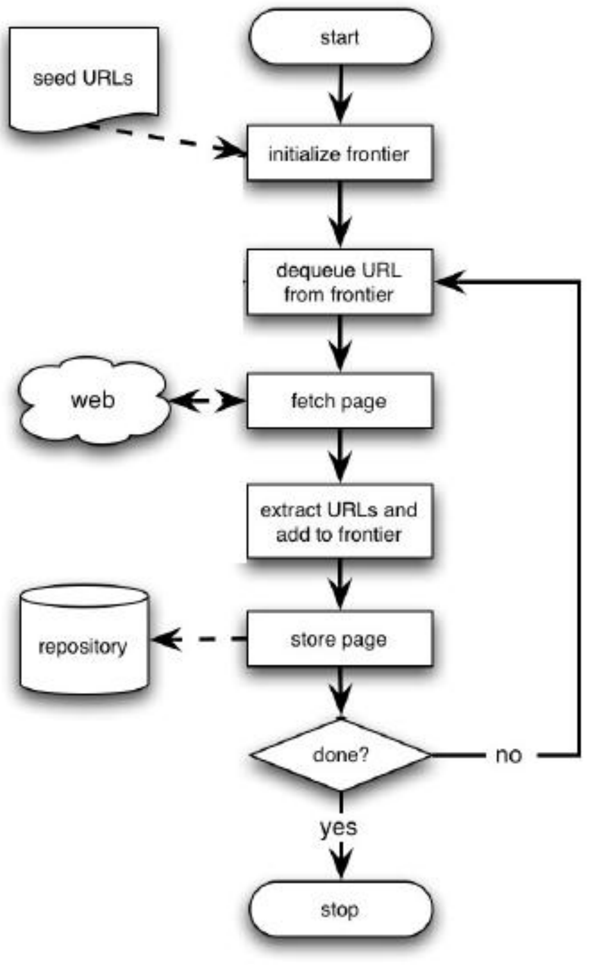


Fig 3.1: Flow Chart of the System.

Code/Design View

The algorithm developed is used by the web crawler to download web pages that are to be indexed by the search engine and based on the algorithm developed above, A source code is writtenin the PHP scripting language that is suited for web development and can be embedded into HTML. The source code is then integrated intothe Apache Server Environment for automatic web search engine fetch sequencing.

The program workability is then test-run in the OSI model layer 7 using Hypertext Transfer Protocol (HTTP).

PHP- Crawler

PHP-Crawler is an open source crawling script based on PHP and MySQL. Created to implement simple as possible local website search it became popular for small websites on shared hosting. Script set is distributed under the terms of BSD (Berkeley Software distribution) License.

REVIEW

Sublime Text is a proprietary cross-platform source code editor with a Python Application Programming Interface (API). It natively supports many programming languages and markup languages, and its functionality can be extended by users with plugins, typically community-built and maintained under free-software licenses

Employing the PHP Scripting Language

PHP is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. PHP code may be embedded into HTML or HTML5 markup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. PHP includes various free and open-source libraries in its source distribution, or uses them in resulting PHP binary builds. PHP is fundamentally an Internet-aware system with built-in modules for accessing File Transfer Protocol (FTP) servers and many database servers, including PostgreSQL, MySQL, Microsoft SQL Server and SQLite (which is an embedded database), LDAP servers, and others. Numerous functions familiar to C programmers, such as those in the stdio family, are available in standard PHP builds.

IV. SOFTWARE DOCUMENTATION AND IMPLEMENTATION

The PHP version 5.6.12scripting language is used to write this software using the Sublime Text Editor. Web crawler is a web based application running on the Apache HTTP Server. The web crawler has four (4) tabs user interface that divides the application into four sections for a convenient user experience.

Test It: This tab takes the user to the main crawl page which provides a form for the user to input the Uniform Resource Locator (URL) which it wants to crawl.

Users: This tab gives an option to view all the users registered in the database of the application.

My Account: This tab provides options for the logged-in user to view his current profile and make changes if need be.

Logout: This tab takes the user to the login / registration page where the user can input his details and login again or register as a new user.

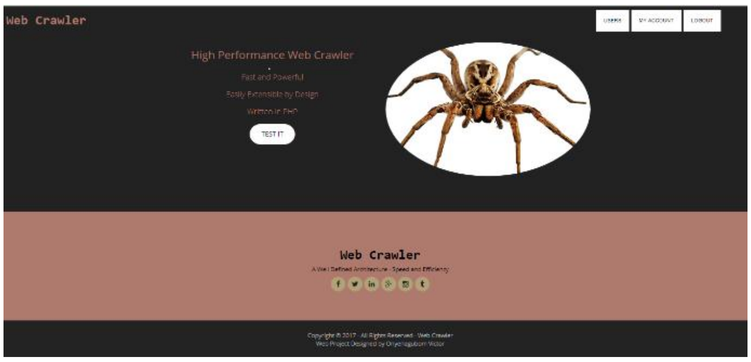
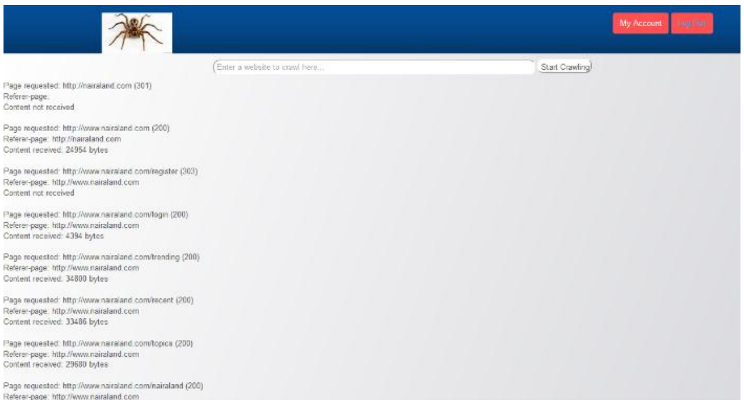
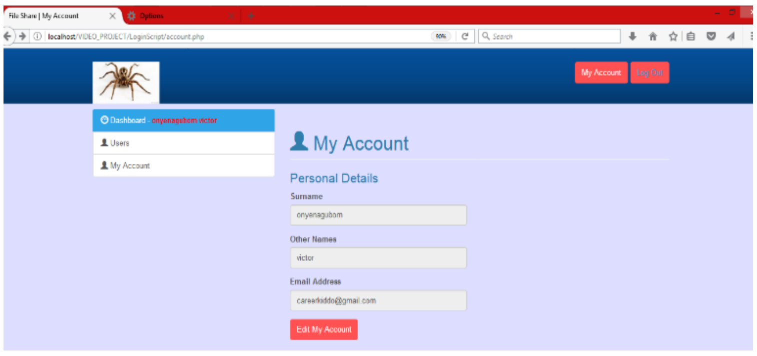

Web Crawler
USERS
MY ACCOUNT
LOGOUT
High Performance Web Crawler
Fast and Powerful
Easily Extensible by Design
Written in PHP
TEST IT
Web Crawler
A Well Defined Architecture - Speed and Efficiency
Copyright © 2017 - All Rights Reserved - Web Crawler
Web Project Designed by Onyenagubom Victor

Fig 4.1: The Index Page after User Login.



```
Page requested: http://nairaland.com (301)
Referer-page:
Content not received

Page requested: http://www.nairaland.com (200)
Referer-page: http://nairaland.com
Content received: 24954 bytes

Page requested: http://www.nairaland.com/register (303)
Referer-page: http://www.nairaland.com
Content not received

Page requested: http://www.nairaland.com/login (200)
Referer-page: http://www.nairaland.com
Content received: 4394 bytes

Page requested: http://www.nairaland.com/trending (200)
Referer-page: http://www.nairaland.com
Content received: 34800 bytes

Page requested: http://www.nairaland.com/recent (200)
Referer-page: http://www.nairaland.com
Content received: 33486 bytes

Page requested: http://www.nairaland.com/topics (200)
Referer-page: http://www.nairaland.com
Content received: 29680 bytes

Page requested: http://www.nairaland.com/nairaland (200)
Referer-page: http://www.nairaland.com
```

Fig 4.2: The Output Design View.



My Account

Personal Details

Surname: onyenagubom

Other Names: victor

Email Address: careerkiddo@gmail.com

Fig 4.3: The Page Showing a Logged In User Profile.

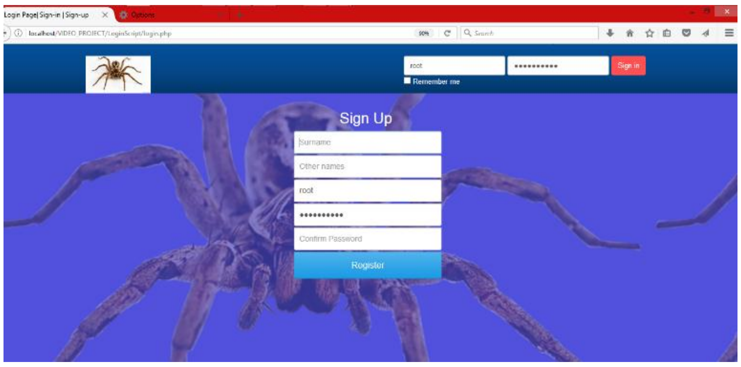
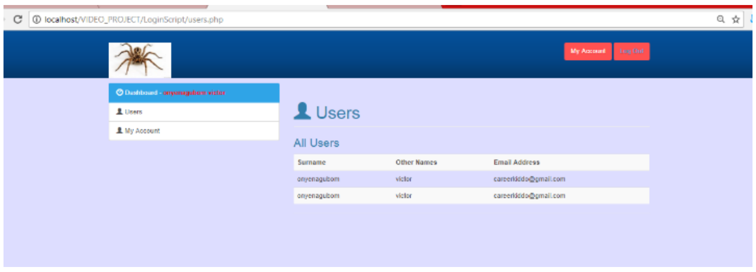


Fig 4.4 Registration Page of the Application


localhost/VIDEO_PROJECT/LoginScript/users.php
My Account
Log Out
Dashboard - onyenagubom victor
Users
My Account
Users
All Users

Surname	Other Names	Email Address
onyenagubom	victor	careerkiddo@gmail.com
onyenagubom	victor	careerkiddo@gmail.com

Fig 4.5: The Page For all Registered Users

A. Operation of the Software

This software simulates a high performance web crawler using the case study of the deployment News Aggregator System that utilizes web crawler technologyin Punch Nigeria Limited contracted to TOUCHSOFTWeb TechnologiesLagos.

For the deployment of such a project the following team players are involved and listed in a chronological order of work responsibility flow for the effective delivery of the job in line with the operational standard of TOUCHSOFT.

- Negotiation with IT Manager
- Approval by the Company's Board of directors
- Allocation of Job to the Project Manager and Supervisors
- Consultancy (Usually by Click Networks limited)
- Release Engineering (Release Management)
- Release Management (Release Management)
- Deployment Coordination

The above responsibilities will attain optimal efficacy with the incorporation of a Web Crawler Software and with cognizance of the fact that development of this prototype is well appreciated.

CONCLUSION

A major open issue for future work is a detailed study of the scalability of the system and the behavior of its components. This could probably be best done by setting up a simulation test bed, consisting of several workstations, that simulates the web using either artificially generated pages or a stored partial snapshot of the web. We are currently considering this and are also looking at test beds for other high-performance networked systems (e.g., large proxy caches). Our main interest is in using the crawler in our research group to look at other challenges in web search technology, and several students are using the system and acquired data in different ways.

REFERENCES

1. Baeza-Yates, R., Castillo, C., Marin, M. and Rodriguez, A. Crawling a Country: Better Strategies than Breadth-First for Web Page OrderinIn Proceedings of the Industrial and Practical Experience track of the 14th conference on World Wide Web, Chiba, Japan. ACM Press, 2005; 864–872.
2. Brin, S. and Page, L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 1998; 30(1-7): 107–117.
3. Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary, 2003. ACMPress, 2003; 280-290.
4. Boldi, P., Santini, M., and Vigna, S. Do your worst to make the best: Par adoxical effects in pagerank incremental computations. In Proceedings of the third workshop on Web Graphs (WAW), volume 3243 of Lecture Notes in Computer Science, 2004; 168-180, Rome, Italy. Springer.
5. Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. Focused crawling using context graphs. In Proceedings of 26th International Conference on Very Large Databases (VLDB), 2000; 527-534, Cairo, Egypt.
6. E. G. Coffman Jr; Zhen Liu; Richard R. Weber. "Optimal robot scheduling for Web search engines". Journal of Scheduling, 1998; 1(1): 1doi:10.1002/(SICI)1099-1425(199806)1:1<15::AIDJOS3> 3.0.CO;2-K.
7. J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In Proc. of the Junghoo Cho; Hector Garcia-Molina. "Estimating frequency of change" .ACM Trans. Internet Technol, 2003; 3(3): 256–290. doi:10.1145/857166.857170. Retrieved 2009-03-22.

8. K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. In 7th Int. World Wide Web Conference, May 1998.
9. Kobayashi, M. & Takeda, K. "Information retrieval on the web" .ACM Computing Surveys. ACM Press, 2000; 32(2): 144–173. doi:10.1145/358923.358934.
10. Lee Giles, The evolution of a crawling strategy for an academic document search engine: whitelists and blacklists, In proceedings of the 3rd Annual ACM Web Science Conference, Evanston, IL, USA, June 2012; 340-343.
11. M. Thelwall; D. Stuart. "Web crawling ethics revisited: Cost, privacy and denial of service". Journal of the American Society for Information Science and Technology, 2006; 57(13): 1771. doi:10.1002/asi.20388.
12. Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In Proceedings of the Tenth Conference on World Wide Web, Hong Kong. Elsevier Science, May 2001; 114–118.
13. Paolo Boldi; Bruno Codenotti; Massimo Santini; SebastianoVigna (2004). "UbiCrawler: a scalable fully distributed Web crawler" (PDF). Software: Practice and Experience. 34 (8): 711–726. doi:10.1002/spe.587. Retrieved 2009-03-23.
14. Paolo Boldi; Massimo Santini; Sebastiano Vigna. "Do Your Worst to Make the Best: Paradoxical Effects in PageRank Incremental Computations" (PDF). Algorithms and Models for the Web-Graph, 2004; 168–180. Retrieved 2009-03-23.
15. Patil, Yugandhara; Patil, Sonal. "Review of Web Crawlers with Specification and Working" (PDF). International Journal of Advanced Research Computer and Communication Engineering, 2016; 5(1).