Indian Journal of Science and Technology, Vol 8(36), DOI: 10.17485/ijst/2015/v8i36/79858, December 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Design of Web Crawler for the Client - Server Technology

Md. Abu Kausar[1*], V. S. Dhaka[1] and Sanjeev Kumar Singh[2]

[1]Department of Computer and System Sciences, Jaipur National University, Jaipur - 302017, Rajasthan, India; kausar4u@gmail.com, vijaypal.dhaka@gmail.com
[2]Department of Mathematics, Galgotias University, Gr. Noida - 201306, Uttar Pradesh, India; sksingh8@gmail.com

Abstract

Search engines store information locally with the purpose of deliver quick, accessible search abilities. This information is collected by Web crawler. Web crawling is necessary for the maintenance of complete and latest web document gathering for a web search tool. The web document modifies its content on regular basis hence it becomes necessary to build up a successful framework which could identify these sorts of changes proficiently in the most minimal scanning time to accomplish this modifications. The essential thought behind designing of such a web crawler is to find high quality web documents within limited time frame. The proposed system works on the Client-Server Technology it reduces the overlap problem and downloads high quality web pages. We can add many web crawler in parallel to download web page in parallel way.

Keywords: Client-Server Technology, Overlap, Search Engine, Web Crawler, Web Page

1. Introduction

Search engines store information locally with the purpose of deliver quick, accessible search abilities. This information is collected by Web crawler. Web crawling is necessary for the maintenance of complete and latest web document gathering for a web search tool. Web crawlers gather web documents from the web with the goal that they can be put away locally and listed via web indexes. Web crawlers must also revisit these web documents periodically in order to keep the local database fresh.

From the development of web crawler, the World Wide Web has been growing at very fast rate, so it's necessary for web crawlers to work efficiently and effectively[1-3]. Some studies[4-8] have been done to approximate the Web size and in spite of the fact that they all report somewhat diverse numbers most concur that over millions of web pages are accessible on Internet. There is no any method or criteria through which we can measure the exact size of WWW, but the approximate size of web can be measured through the records kept up by some popular search engines[9].

The main problem related with any search engine is that it is not possible to index the whole Web, as the Web is very dynamic and continuously growing. So refreshing and indexing of such web documents within reasonable time is a challenging task for every search engine. Web crawlers work in a situation where independent modifications are made to target web documents. To keep up repository consistent at search engine side, web crawlers must revisit web documents on periodic basis after they have been initially crawled.

Search engines need to perform various web crawlers in parallel to decrease the download time of web documents, while working in parallel the accompanying concerns that must be addressed[10].

- Overlap: When several web crawler keep running in parallel to download web documents, it is feasible that different crawler running on parallel basis may download the similar web document several times because one web crawler may not know that a new crawler has previously downloaded the similar web documents or not. Nowadays a lots of organization also copy their web pages on many web servers to keep away from

*Author for correspondence

server corruptions in such case a web crawler may also downloads same web documents. Such multiple downloads should be reduced to save network bandwidth and increase the web crawler's efficiency.

- Quality: The main job of web crawler is to download significant web pages first to maximize the quality of the downloaded web documents. Thus, to download such appropriate web pages, multiple web crawlers which are running in parallel must have the overall image of all things considered downloaded web pages so that excess as far as duplicate web documents may be stayed away from.
- Communication bandwidth: To increase the quality of the retrieved web documents or avoid the overlapping of retrieved web documents, the crawling processes need to communicate periodically to coordinate with one another. However, this communication may grow considerably as the number of crawling processes increases.

Thus there is a need to develop a parallel web crawler which reduces the overlap problem and also downloads high quality web pages.

In this paper, "Design of a Web Crawler for the Client-Server Technology" is designed that addresses the above issues. It reduces the overlap problem, downloads high quality web pages and also avoids unnecessary communications among downloaded web pages. The architecture of proposed system is explained in following section.

2. Related Work

The file of getting data from the web is not just the work of web crawler which is extracted however it additionally handle different issues, for example, the allotment of sites which is to be retrieved, system use and carrying the downloaded web documents together with before hand retrieved information, replication of data and coordination of outcomes.

Author[10] talks about distinctive conditions and choices for parallelization of crawling technique. The crawler is all around circulated or controlled centrally. Crawler is intended to keep away from cover of web documents that are downloaded while taking consideration the network load. The authors depict the element of a web crawler that takes the most significant web documents prior to others. This procedure is important for web crawler which works on parallel basis as all the crawling process just focus on neighbourhood information for building the web documents significant. Authors additionally expressed that distributed crawlers are advantageous than multithreaded web crawlers in light of viability, versatility and quantity. In the event that decrease of system load and supply of system are finished then parallel web crawler can surrender top notch web documents.

Mercator is a web crawler which is extensible and versatile and is presently turned into the Altavista[11] search engine. [12]Comment on execution concern which is familiar for developing of parallel web crawler, which can goes down execution. Author discuss about benefits and faults of different coordination systems and appraisal criteria. Creators audit the work with incredible presentation data. To put it plainly, creators concur that the overhead of correspondence does not increment as more web crawlers are added, when more nodes are included then framework's throughput increases and the element of the framework also, i.e. the ability to get important web documents first.

Fetterly et al.[13]clarifies a most important analysis for calculating the change rate of web pages over a long period of time. Author download around 151 million web documents once consistently over a time of eleven weeks. Creator note down most essential data about all downloaded web documents. Shkapenyuk et al.[14] yield information which is identified to create by[11]. Their course of action uses one gadget for each measured work. To avoid bottlenecks while enhance, they needed to introduce more gadgets for precise work. In this manner, to append additional web crawler the quantity of web crawler opened up with a more unmistakable curve. The correspondence of system load increased without a doubt brought on by raised coordination work between machines with similar kind of works.

Papapetrou et al.[15] connected a framework for get web documents from servers close to them by the utilization of all around circulated crawlers. The structure is adaptable to breakdowns however it has long adaptability point and the characteristic is unsafe for data. At the point when crawler fails at that point, the documents which is downloading are shifted to other crawlers for downloading the web document which are not downloaded. A large number of overlap of information take place because of this outcome. For pick the following web crawler a genuine pulse convention is compulsory. In addition, a crawler nearer to different web server may be over-load where a web server may be inactive at a little bit more distance.

Cho et al.[16], describe the importance of URL rearranging in the crawl frontier. If web documents exis-

tent in the crawl frontier and is related with alternate kind of web documents that are to be downloaded, a few quantity of web documents that are related from it makes intelligence to visit it before others. Page Rank is utilized as an overwhelming metric for URL ordering and assembled three models to assess web crawlers. Authors presume that Page Rank is a superior metric and web documents with a high Page Rank or ones with different back links are required first.

Authors[17], learned about constructing an incremental web crawler. They describe the periodic web crawler as the web crawler who retrieve the web documents until the repository has a large number of web documents, and quits retrieving web documents. When it is necessary to update the collection, the web crawler makes a new gathering of web pages with the same process defined above, and at that moment replaces the old gathering with this new one. On the other hand, an incremental web crawler downloads and updates web documents in the repository after a desired number of web documents are downloaded and stored incrementally. By this incremental update, web crawler update present web documents and exchange web documents which are not important with new and more relevant web documents. Authors describe Poisson process for check the rate of changes in the web documents. A Poisson procedure is periodically connected to show a game plan of arbitrary occasions that happen individually with fixed rate of time. Authors disclose two techniques to keep up the database. In beginning one, gathering of different copies of web documents are put away in the repository in which they were originate during crawling and in the second one those duplicates of web documents are stored which is generally new. Therefore one needs to keeps a record of when and the amount of the time the website pages changed. In this paper the creators infer that an incremental web crawler can produces fresh copies of web documents more quickly and keep up the repository fresher than periodic web crawler.

Kausar et al.[18,19] proposed a Model for Web Crawling which is based on Java Aglets. Web Crawling based on Mobile Agent will produce high quality web documents. The process of crawling will transfer to web server for start downloading.

Vinit et al.[20] proposed a search engine based on perception which returns results according to the user opinion. To accomplish semantic searching, a knowledge base is built which stores knowledge as predicates. To extract knowledge from knowledge base, decision theory is used.

3. Architecture of Proposed Approach

The fundamental thought behind designing of such a web crawler is to find high quality web pages within limited time. The proposed system is based on client server based architecture. The proposed architecture is carried out in VB.NET to crawl the web documents and crawled information is stored in the central repository[21,22]. The complete building design of a proposed framework is appeared in Figure 1.

The whole process consists of following main components. These are:

3.1 URL Dispatcher.
3.2 Ranking Module.
3.3 URL Distributor.
3.4 Crawlers.
3.5 URL Buffer.
3.6 Change Detection.
3.7 Indexer.
3.8 Repository.

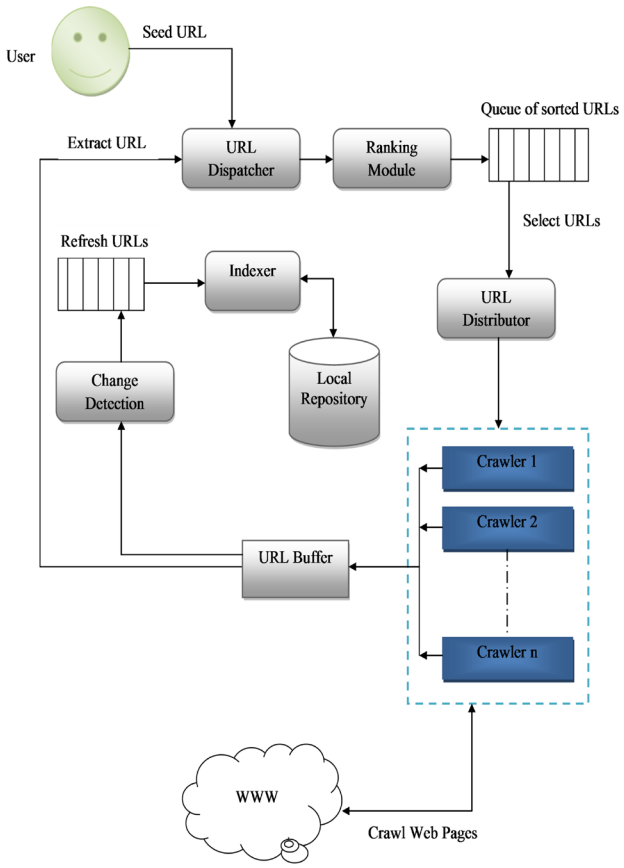

Figure 1. Work flow of the traditional crawling system.

3.1 URL Dispatcher

The URL dispatcher is begun by initial URL. In starting sorted URL queues are unfilled and the seed URL got from client is put away in sorted URLs queue. URL distributor picks the URL from queue of sorted URLs and allocates it to the web crawler for retrieve the web documents. After retrieving the web documents from the World Wide Web, the web crawler separate its contents to take out the embedded URLs present in it and stores the web documents and related URLs in URL buffer. Every last URL, recovered by the URL dispatcher is confirmed from local repository before putting it into the sorted queue, to distinguish whether it has already downloaded or not. If the URL and its corresponding web documents are existent in the database then the downloaded URL is excluded and not saved on the repository. URL dispatcher spares all new URLs in the line of sorted URLs in the request they are recovered from the URL buffer. This entire process is rehashed till the line of sorted URLs turn into blank.

3.2 Ranking Module

The ranking module uses a combination of related keywords present inside of web documents and their back-connections to rank the web pages[23]. The web page and its back-connections are later organised in view of the rank obtained. The ranking module works in the following two stages: in the first stage it calculates the related keywords present in a web document and its back-connections, and in the later stage, it positions the web document and its back-connections. Therefore, the two staged ranking framework recognizes the most appropriate web documents for a given client inquiry. Working of Ranking Module is given in Figure 2.

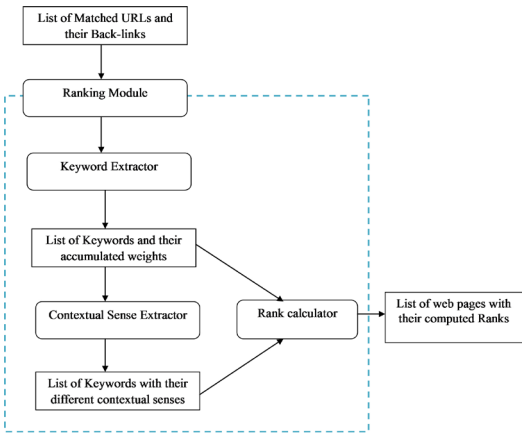

Figure 2. Working of Ranking Module.

The ranking module takes the list of recovered web documents and their back-connections as data from the enquiry module. The keyword extractor extracts the list of keywords present within web document, their relative recurrence at several tags, and calculates the relative collected weight of every keyword. Moreover, the contextual sense extractor extracts the related keywords present within web document. Finally, the rank calculator computes the rank of the web document and their back-connections in light of the accumulated weights and their related keywords. The most elevated ranked web documents and back-links for a user query are presented as the highest result.

3.3 URL Distributor

URL distributor chooses the URL from the sorted queue and assigns it to client crawler according to the equation (1), this process is repeated till the queue of sorted URLs gets to be vacant.

We assume that f_1, f_2, ------f_t are the factors to be measured, and for a crawler their equivalent calculated value are v_1, v_2, --------- v_t ($0 \le v_i \le 1$), and the weights of the factors are w_1, w_2, ---------- w_t; then the total calculated value can be calculated by given below formula; finally, rank the crawlers according to the total calculated value, and select the best crawler for downloading.

$$f(x) = \sum_{i=1}^{t} v_i w_i \qquad (1)$$

3.4 Crawlers

The functioning of Web crawler is starting with a set of URLs which is known as seed URLs. Crawler download the web pages from the seed URLs and extract the new URLs that are available in the retrieved web documents. The downloaded web documents are saved and well indexed on the repository so that by the assistance of these indexes they can later be recovered as and when needed. The links that are retrieved from the downloaded web documents are confirmed from repository to know whether their related web pages are previously downloaded or not. If the related web page are not downloaded, then the URLs are again allotted to crawlers for further downloading. The above same procedure is reiterated till no further links are available for downloading. To complete the target web crawler download millions of web pages on daily basis. The numbers of crawlers supported

by the architecture is not fixed. It may be increase or decrease depending on accessibility of assets and the size of real requirement. Figure 3 illustrates the architecture of crawler.

The web crawler have the capacity to handle a wide range of web documents accessible on the WWW. Presently technology is developing at very fast rate, the web is no more restricted to basic HTML pages it consists of different varieties of web pages which are utilized to show dynamic content and constantly changing designs. The crawlers are applied in a manner that they are capable to perfectly break down the web document contents and also handle every kind of web document in an effective way. Before beginning the real downloading of web documents a web crawler checks for its genuine presence on the Web by the remote server response. In the event that there is no reaction from the web server or the web document is restricted to be visited from specific networks as indicated by the corresponding robot.txt file, it quickly rejects the web page without waiting for further comebacks in order to save web resources for quicker crawling.

3.5 URL Buffer

Every URL retrieved by the URL buffer is checked from database before placing it in the database to know whether it has previously downloaded or not. If it is found that the URL is previously downloaded and the matching web page accessible in the repository then the downloaded web page is discarded and not saved in the repository. By doing this, URL dispatcher verifies that the same URL is not used many times to download its matching web pages from web and so it helps to save network bandwidth.

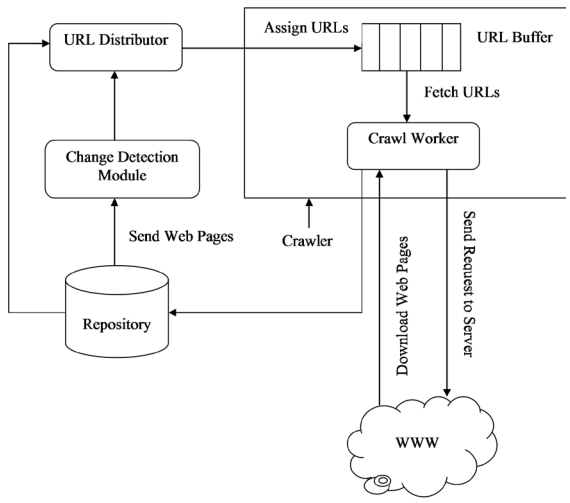

Figure 3. Architecture of Crawler.

3.6 Change Detection

Web page change detection is computed by matching the recently retrieved web document to a formerly retrieved variant saved on database. Change detection works on the basis of calculating the Hash value of two HTML documents. The web document parts that are not fully related are considered to have been changed.

The web page change detection technique calculates checksum for the whole web document. While upgrading the web document in the repository, comparison between the checksums of both versions of web documents are made and if there should be an occurrence of any distinction, it is satisfied that the web document at web server site has been altered as compared to the local copy of web document keep up in the repository at search engine side.

3.7 Indexer

Indexer retrieves web pages and relating URLs from URL buffer. The web pages are put away in database and relating index is made. The indexes generated for the web pages are normally be made up of checksum values of web pages, keywords present in the web pages, address of web pages in repository, corresponding URL of web pages through which the web pages was downloaded and some other information. All these indexed information for web pages help to generate suitable result when users enter search query using search engine.

3.8 Repository

It is basically a centrally indexed database of whole web pages which are maintained by server. It later may be utilized by search engines to answer the user's inquiries. The proposed method keep up index of whole web documents in the database along with other essential information like checksum values of web documents, keywords existent in the web documents, address of web pages in repository, corresponding URL of web pages through which the web pages was downloaded etc. The proficiency by which the repository is proficient affects the results of search. Since users require most suitable results for a given query the database and relating data maintained for these web documents play a most important character as well as ranking algorithm to deliver the outcome. It is realized that every search engine's database

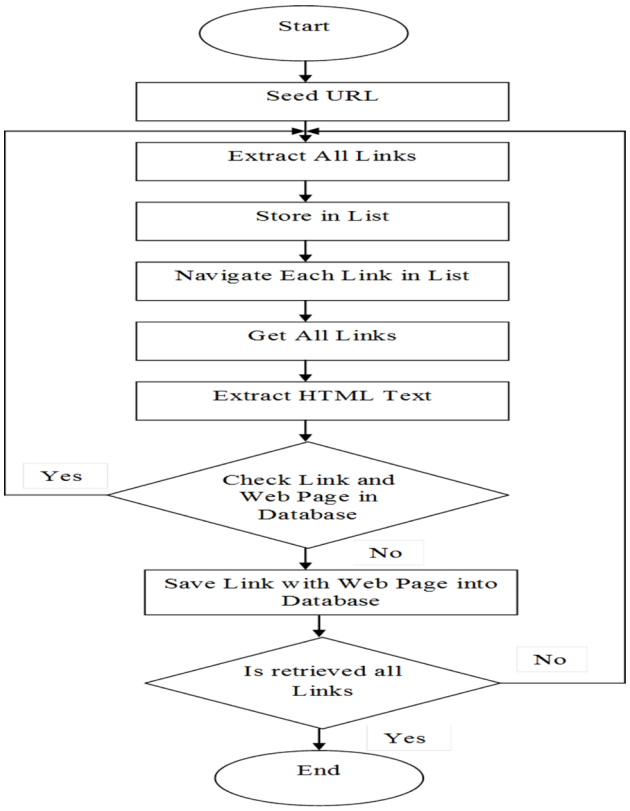

Figure 4. Proposed Crawling Process.

contains billions of web pages. The clients need answer in fast time periods. So, there is a requirement for effective indexing of the database.

4. Complete Working of Proposed Approach

The complete working of all components of the proposed architecture for web crawler for client server technology, discussed in this chapter, may be represented by following flowchart. Figure 4 illustrate the complete working of proposed web crawler.

5. Conclusion

In this paper, design of web crawler for the client - server technology has been proposed. The basic idea behind designing of such a web crawler is to find high quality web documents within limited time. The proposed system is based on client server based architecture. The detail about the main components and their whole process has been discussed in this paper.

6. References

1. Brin S, Page L. The Anatomy of a Large-Scale Hyper textual Web Search Engine. In: Enslow PH Jr, Ellis A, editors. Computer Networks. 2012; 56(18):3825–33.
2. Heydon A, Najork M. Mercator: A Scalable, Extensible Web Crawler. World Wide Web. 1999; 2(4):219–29.
3. Shkapenyuk V, Suel T. Design and Implementation of a High-Performance Distributed Web Crawler. Proceedings of the Eighteenth International Conference on Data Engineering (ICDE'02), San Jose, CA. 2002. p. 357–68.
4. Cho A, Garcia-Molina Hector J, Paepcke A, Raghvan S. Searching the Web. ACM Transactions on Internet Technology. 2001; 1(1):2–43.
5. Bar-Yossef Z, Berg A, Chien S, Weitz JFD. Approximating aggregate queries about web pages via random walks. Proceedings of the 26th International Conference on Very Large Data Bases. 2000. p. 1–10.
6. Bharat K, Border A. Mirror, mirror on the web: A study of host pairs with replicated content. Computer Networks. 1999; 31(11-16):1579–90.
7. Bharat K, Border A, Henzinger M, Kumar P, Venkatasubramanian S. The connectivity server: fast access to linkage information on the Web. Computer Network ISDN Systems. 1998; 30(1-7):469–77.
8. Lawrence S, Giles C. Searching the World Wide Web. Science. 1998; 280(5360):98–100.
9. Infoseek. 2008 Mar. Available from: www.infoseek.co.jp
10. Cho J, Hector G-H. Parallel Crawlers. Proceedings of the 11th International Conference on World Wide Web WWW '02. Honolulu, Hawaii, USA. ACM Press. 2002. p.124–35.
11. Altavista. 2008 Mar. Available from: www.altavista.com
12. Heydon A, Najork M. Mercator: A scalable, extensible web crawler. World Wide Web. 1999; 2(4):219–29.
13. Fetterly D, Manasse M, Najork M, Wiener J. A large-scale study of the evolution of web pages. Proceedings of the Twelfth International Conference on World Wide Web, Budapest, Hungary. 2003. p. 669–78.
14. Shkapenyuk V, Suel T. Design and implementation of a high-performance distributed Web crawler. Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, CA. 2002. p. 357–68.
15. Papapetrou O, Samaras G. Minimizing the Network Distance in Distributed Web Crawling. International Conference on Confederated, Coopis, DOA, and ODBASE, 2004. p. 581–96.
16. Cho J, Molina HG, Page L. Efficient Crawling Through URL Ordering. Computer Networks and ISDN Systems. 1998; 30(1-7):161–72.
17. Cho J, Molina HG. The Evolution of the Web and Implications for an incremental Crawler. Proceedings of 26th International Conference on Very Large Databases (VLDB). 2000. p. 200–9.

18. Kausar MA, Dhaka VS, Singh SK. Web Crawler Based on Mobile Agent and Java Aglets. International Journal of Information Technology and Computer Science. 2013; 5(10):85–91.
19. Abu Kausar M, Dhaka VS, Singh SK. An Effective Parallel Web Crawler based on Mobile Agent and Incremental Crawling. Journal of Industrial and Intelligent Information. 2013; 1(2):86–90.
20. Kumar V, Singhal N, Dixit A, Sharma AK. A Novel Architecture of Perception oriented Web Search Engine Based on Decision Theory. Indian Journal of Science and Technology. 2015; 8(7):635–41.
21. Abu Kausar M, Dhaka VS, Singh SK. Implementation of Parallel Web Crawler through .NET Technology. International Journal of Modern Education and Computer Science (IJMECS), Hong Kong. 2014; 6(8):59–65.
22. Abu Kausar M, Dhaka VS, Singh SK. A novel web page change detection approach using Sql Server. International Journal of Modern Education and Computer Science (IJMECS), Hong Kong. 2015; 7(9):36–43.
23. Gupta P, Singh SK, Yadav D, Sharma AK. An Improved Approach to Ranking Web Documents. Journal of Information Process System. 2013; 9(2):217–36.