An Efficient Technique to Reduce Network Load during Web Crawling

Kompal Aggarwal
Govt. College, Chhhachhrauli

Abstract:
As the size of the web continues to grow, searching it for useful information has become increasingly difficult. Sufficient of current internet traffic and bandwidth consumption are due to the web crawlers that retrieve pages for indexing by the different search engines. These crawlers also cause load on the remote server by using its CPU cycles and memory.

Moreover, due to the dynamic nature of the web, it becomes very difficult for a search engine to provide fresh information to the user. People upload the web pages and updating the new web pages very frequently. There is a frequent change in the content of the web page hence it become necessary to develop an efficient system which could detect these changes efficiently and in the minimum browsing time. The mobile crawlers filter out pages that are not modified since last crawl before sending them to the search engine for indexing purpose and to achieve this the old web page is compared with the new web page. The web page change detection system can be implemented by using various Tools or Algorithms.

Keywords: Search engine, mobile crawler, Network load.

1. Introduction
The Search engine create and maintain data base for indexing purpose to process search related queries. The information collected by web crawlers is used for creating the indexes used by search engines. The traditional web crawler uses link and download methodology. The mobile web crawler can transfer itself and reside in the memory of the remote server. The great access to a server may overload a server and may crash the server. To improve the efficiency of a search engine, we must develop an effective crawling strategy to save bandwidth and maintain the quality of web searches. For efficient crawling we must formulate a methodology that find out the changes pages and revisit pages based on selection. The crawling strategy must be able to decide about page change without completely analyzing the page for any changes. This paper proposes various methods which intelligently decide about change in a page using various parameters and skip unchanged pages.

2. Related Work
Brandman et al. [12] have studied the idea of how to make web servers more crawler friendly through that web servers export meta-data archives describing their content, so that crawlers can efficiently create and maintain large, “fresh” repositories. This meta-data includes the last modified date and size for each available file. This approach reduces the network bandwidth by sending only the modified pages after last crawling date, but in the same time it provides the search engine’s index with significant changes (content changes) and insignificant changes (structure and attributes changes) together. These insignificant changes results in network overload and wasted resources for the search engine through re-crawling and re-indexing the web pages for insignificant changes. The authors here concern only with the search engine’s index freshness and ignore the distributed crawling and indexing.

Yadav et al. [13,14] have proposed checksum (hash value) based content level change detection. At the time of page crawling, only comparison will be made to the text code of that page. The main drawback of this technique is that if any change in that value is detected for the actual copy on the web as compared to the local copy, regardless it is significant or not, the page will be refreshed or re-crawled. Hence this technique results in network overload and wasted resources for the search engine.
Artail and Abi-Aad [15] have proposed a web page change detection approach based on restricting the similarity computations between two versions of a given web page to the nodes with the same HTML tag type. Before performing the similarity computations, the HTML web page is transformed into an XML-like structure in which a node corresponds to an open-closed HTML tag. This tree structure uses a lot of storage space as well as causes a lot of inconvenience at time of refresh, as the tree structure has to be compared. Also this approach works only on the page types that can be transformed into an XML-like structure such as HTML pages.

Bal et al. [16,17] have proposed a novel indexing system based on mobile agents, which can filter out the HTML pages that have not been modified since last crawl through two web page change detection methods. The first method is the comparison of page sizes of web pages at the time of page change detection. The second one uses the last modification date of web pages. These methods have the same drawback like the hashing method in above discussed related work as any insignificant change will change both the page size as well as its last modification date. This leads to overloading the search engine with processing web pages that will not change the index.

[14] Describes web page change detection system based on Signature of Node corresponds to HTML pages. In this proposal first HTML document is filtered into XML structure document and then transforms XML pages to trees using DOM. They assume that only text nodes can be changed elements and attributes nodes can not be considered. The node signature comparison algorithm is developed to compare the trees of old web page and modified web page to find the changes in the web page. This system highlights changes of content i.e. deletion and addition of text and attribute changes i.e. font change, caption change, color change etc. and highlights the changed part in red color and displays to the user. This algorithm gets result faster as it does not search the sub tree if that sub tree does not have any changes.

This algorithm follow a top down approach it start with the root node by comparing the signature values of each node to its corresponding node in the two trees. To reduce the number of comparison we use node signature. Algorithm assigns the hash value to all child nodes in the trees the child nodes are basically the text nodes. Signature is mainly the function of the hash value calculated from the contents of the node. Signature of child node is basically the summation of its entire child node signature except the leaf node.

[3] Proposed an architecture that concentrates more over page selection policy and page revisit policy. The three-step algorithm for page refreshment serves the purpose. The first layer contributes to decision of page relevance using two methods.

The second layer checks for whether the structure of a web page has been changed or not, the text content has been altered or whether an image is changed. Also a minor variation to the method of prioritizing URLs on the basis of forward link count has been discussed to accommodate the purpose of frequency of update. And finally, the third layer helps to update the URL repository.

[18] proposed an algorithm which will find structural as well as the content change. The hash based algorithm is to be used to detect the changes. First a web page is searched in which changes had to be detected. Then the tree will be designed for that web page and then the two trees are compared by the tag values assigned to each node. Before finding the changes URL of a particular web page to be searched. A crawler has been designed which will save the HTML code of the page.

Another algorithm is designed which will develop the trees corresponding to the pages. Algorithm is mainly implemented in C# and uses the inbuilt classes and methods of C#. Tree development algorithm is mainly based on tags extraction and then developing the tree while the change detection algorithm is based on assigning the hash value to each node which we have calculated by giving each tag a number and the level number where the child is placed in the tree. Hash value is mainly assigned to the leaf nodes in the tree and for the non leaf nodes the tag value will be assigned which is the sum of hash value of its child. Tag value is assigned in a bottom up manner and uses them to find the changes. If the tag value of a node changes then we can say that there is a change.

[19] discussed Page refresh approach to derive certain parameters, which can help in deriving the fact that whether the page is fresh or same as the page present in web repository. These parameters are found out at the time of page parsing. Whenever
a client fetches a URL which is already present in the data repository, it compares the new web page with the existing one with the help of matching parameters. If matching parameters shows differences between two copies it update the repository with the new copy otherwise it discard the new copy & continue with the older one. According to the previous approach the whole page is parsed & the formula is applied on the whole page. But as per our approach rather than comparing whole page we just compare the documents till the first unmatched value. Because if a single value is found to be different, page is declared to update & the older copy will be replaced by the fresh copy.

They used UNICODE which has developed a coding system that includes every written alphabet in existence. They assign a code to the text content appeared in a web page. They did the process of assigning code word by word. Whenever difference is found between two words we came to the conclusion that page has been updated otherwise comparison is made till the end of the web page. The formula for text coding is as follows: \( \Sigma (\text{Character frequency}) \times \text{UNICODE} \)

Distinct Symbol Count

Where Frequency is the sum of the occurrence of a character in a web page. UNICODE symbol defines the code of a character. Distinct Symbol Count is the count of different symbols appearing in the page.

In the approach proposed by [8], The crawler manager employs a page change matrix (PCM) of records. The PCM contains a record for each URL, which may be crawled by the mobile web crawler. TABLE 1.0 shows the PCM containing a number of parameters.

<table>
<thead>
<tr>
<th>YPA</th>
<th>( \Delta m \pi )</th>
<th>( \Delta M \Pi )</th>
<th>NPC</th>
<th>PCF</th>
</tr>
</thead>
</table>

The columns in the PCM represent parameters such as DMP, DMR, NPC, and PCF. DMP and DMR fields specify the previous date of modification and the recent modification date of a page. The crawler manager updates these fields, whenever a page change is detected and the page is downloaded from the web server. NPC represents the number of times the page belonging to a particular URL has changed. PCF specifies the page change factor, which is calculated as follows: \( \text{PCF} = \frac{\tau K}{\text{TM}} \)

Where \( \tau K \) is calculated as the time interval between previous change and recent change in a page. The TM specifies the total time duration over which a page is monitored for page change.

[17] proposed a document index based change detection technique and distributed indexing using mobile agents. The proposed system can considerably reduce the network traffic and the computational load on the search engine side and keep its index up-to-date with significant changes.

The key idea of their architecture is based on the Master-Slave agent design pattern. On the Master-Slave design pattern, a master agent delegates a task to be done on a given agency to a slave agent, in order to continue executing other tasks that cannot be interrupted. The slave agent visits the indicated agency, where it accomplishes the task and then returns to the source agency with the results. The master agent receives the results from the slave agent. Then, the slave one destroys itself.

The main contribution of the proposed system is the creation of document index of web pages at the web server side. This proposed approach has three advantages: First, it is used as a change detection technique where it is more robust to non-significant web page changes than other existing change detection techniques that depend on page size, last modification date and hash value. Second, the document index of changed pages is returned to the search engine instead of the pages themselves. Therefore, it reduces the network load results from crawling. Third, it reduces the computational load at the search engine side because the document indices of the web sites are already created at the web server and the search engine has to only create the inverted indices.

Their attention toward content change of a web page, not the structural change. So it is important firstly before starting re-crawling process to determine if the web page which already crawled before has been changed or not, either the change structural or content. This filtration question saves web server’s crawling time, indexing time, CPU cycles and memory. If the web page has not changed since the last crawling cycle, then no need to re-crawl and index it again. We determine that through the last modification date (LMD) of a web page which is saved in the meta-data of the web
page. If the LMD has changed, then it indicates two probabilities, either a web page has structural change or content change. SA starts only re-crawling process for only crawled web pages with changed LMD. SA compares the web page’s old document index which is saved on the web server from last crawling with the new one. If the indices of a web page are similar, then this web page has an insignificant change and hence no need to transmit its document index again over the network to the search engine. But if the indices are different, it means that a web page has actually changed in content and its document index should be transmitted over the network to MA. MA receives the updated document index of a web page and replaces it with the old one in search engine’s database. Afterward, the search engine creates the inverted index from the up-to-date document indices. The following information are collected and stored after each crawl during the experiment:

- Number of pages that were added/deleted/modified after the last crawl.
- The parameters responsible for change detection—last modification date, page size in bytes, hash value, keywords count (page index).
- The number of bytes retrieved directly by the crawler.

[10] In the literature, a Poisson process is often used to model the change of a web page. They believe that it is a good model because a Poisson process models a sequence of random events that happens independently with fixed rate over time. Also, we make the assumption that web page change at a uniform rate. Thus, the average rate of change \( \lambda \) is uniform. From literature survey it is concluded that:

- Most of the approaches involve user participation in larger extent in order to get the fresh and more relevant information. It is the user who chooses which pages the crawler must update. The pages, which are not very popular, are not updated frequently by the crawler.
- Moreover, the major difficulty is to stimulate the different behaviour of users. It is obvious that a novice user will not have the same behavior than a person who has advanced computer skills.

- As web pages are changing at very different rates, the crawler needs to carefully decide which pages to revisit and which pages to skip in order to achieve high “freshness” of pages.

Revisiting the frequently changing web pages cannot obviously improve the effect of search engine. We should focus the resources on the web pages changing not so quickly. However, a problem is still existent. Page update frequency has been modeled with statistical functions such as Poisson distributions. Actually these models are not exact. There are numerous of unchanged web pages during the updating period. And re-crawling unmodified web pages implies a cost in terms of network bandwidth and resource usage. Consuming little for a single page, it becomes considerable on large scale. Under the typical refreshing strategy the crawler revisits all pages at the same frequency regardless of how often they change. In fact, a large number of pages change very slowly.

3. Problem Statement

Dynamically changing nature of web requires that mobile web crawlers must be able to intelligently decide about new pages and the changes in already crawled pages. A mobile web crawler must revisit the pages, which had been crawled and downloaded earlier, to keep the indexes updated at search engine. As the web pages change in an undefined pattern, the mobile web crawler must be able to decide intelligently about which pages are to revisit and which pages are to skip. The ability of a mobile web crawler to decide carefully about the pages to be transferred through the network, results in significant reduction of load on the resources of the web server and underlying network. To address the problem as stated above, an efficient mechanism for mobile web crawler to revisit the already crawled pages and select only changed pages for downloading to the search engine is to be developed.

4. Classification of changes in a web page

Web page changes can be classified as follows:

1. Structural Changes
2. Content Or Semantic Changes
3. Presentation Or Cosmetic changes
4. Behavioural Changes
Structural Changes
These changes occur when ever a tag is added or deleted in a web page i.e. addition or deletion of a tag causes structural change\[11\] in a web page. Sometimes the addition/deletion/modification of a link also causes a structural change. These types of changes are important to find as they are not visually perceptible.

Content or Semantic Changes
These changes occur whenever the content of a web page changes according to the reader point of view. [11] A stock trader may be interested to know the changed status of the market or the current price of the share. He is interested in the current or the changed status of the market and not in the old price or the old market status.

Presentation or Cosmetic Change
These Types of changes Occur whenever the appearance of a web page is modified but the content of a web page remains the same [11]. For example: with the changes in tags the appearance of a page may change without change in the content of a page.

Behavioural Changes
Behavioural changes refer to changes in the active components which are present in a document[11]. For example, web pages may contain scripts, applets etc as active components. When such hidden components change, the behaviour of the document gets changed. However, it is difficult to catch such changes especially when the codes of these active components are hidden in other files.

5. General Architecture
The general architecture is shown below in the figure. This architecture contains the comparator module where various algorithms can be applied to compare the two web pages. [9].This architecture follow a major path from the start state to the end state these states are denoted by the ellipse. It corresponds to the steps taken to process a user request. The user will input the old web page and modified web pages in which the changes is to be detected.

The input module then asks the crawler to fetch the two pages. The old web page will be fetched from archive and the new web page will be fetched from the site. These pages will be saved in page report archive. These pages will be sending to the tree builder module via manager. Tree builder module will design the trees corresponding to the old and the new web page. These trees will be passing on to the comparator module. In this module comparison algorithm is applied to find the difference between the two web pages. Then the result will be given to the presentation module. Presentation module prepares a report from these results and save it in page report archive. Notification centre will assemble all the reports related to that web page compile them and send it to the user. There is an inverse tree builder module which will generate the page from the tree and give the pages to the browser to show the result.

Issues with traditional search engine design: Web crawlers consume significant resources to continuously crawl web servers and retrieve newly added/recently modified recently modified pages. This section presents some of the main issues related with traditional crawling mechanisms.
LOAD CONSUMPTION
A traditional crawler attempts to download all pages recursively from a target server. The continuous growth of web further increases the load on overall network.

LOCAL ANALYSIS
The traditional crawlers download all pages to the local site before a search engine issues queries for fetching relevant information and carrying further analysis. All pages transferred to the local site may not be useful and are discarded in the process of indexing, this results in unnecessary burden on the performance of the search engine and significant wastage in network bandwidth.

INDEXATION
The search engines need to continuously refresh their indices to keep updated information about changed pages. The traditional crawlers cannot analyze the page content prior to page download. This puts heavy load on search engines to revisit already visited pages and download all those pages before analyzing and starting the process of indexing.

The major problem with the traditional crawling techniques is that there is no way to determine at the remote site, the pages that are actually modified after the last crawl and needs to be re-indexed. This will result in wastage of resources in indexing the pages that have already been indexed by the search engine. The studies of [6, 7, 12] introduced the mobile crawling techniques to overcome the problem of traditional crawling techniques.

The major advantages of a mobile crawler over the traditional crawler are as follows:

i) The mobile crawlers can access Web pages locally with respect to the server. This saves network bandwidth by eliminating request/response messages used for data retrieval.

ii) The mobile crawlers can select only the relevant pages before transmitting them over the network. This saves network bandwidth by discarding irrelevant information directly at the data source.

iii) The mobile crawlers can compress the content of Web pages before transmitting them over the network. This saves network bandwidth by reducing the size of the retrieved data.

6. CONCLUSION
This study is to develop an efficient web page change detection system which will detect the changes in a web page.

The proposed algorithm extracts changes between different versions of web pages. The algorithm is able to make out structural as well as content based changes.

6. REFERENCES
[6]. Available at: Mind-it, http://www.netmind.com
[7]. Leonardi E., Sourav Bhownick S., “Detecting Content Changes on Ordered XML Documents Using Relational Databases”.
[10].vipul Sharma , Mukesh Kumar, Renu Vig, a hybrid revisit policy for web search, journal of advances in information technology, vol. 3, no. 1, February 2012.


