Crawling the Web Surface Databases

Vidushi Singhal
Asstt Prof
MRIU, Faridabad

Sachin Sharma
Asstt Prof
MRIU, Faridabad

ABSTRACT

The World Wide Web is growing at a rapid rate. A web crawler is a computer program which independently browses the World Wide Web. The size of web as on February 2007 was 29 billion pages. One of the most important uses of web page is in indexing purpose and keeping web pages up to date which can be used by search engine to serve the end user queries. Web is dynamic in nature; hence we need to update the web pages constantly. In this paper, we put forward a technique to update a page stored in web repository. This paper put forward an efficient method to refresh a page. We are proposing two methods for refreshing the page by comparing the page structure. First method compares the page structure with the help of tags used in it. And second method creates a document tree compare structures of pages.

Keywords: Web Crawler, WWW, Spidering, Search Engine, Surface Web, Deep Web, Document Tree Structure

1. INTRODUCTION

1.1 Definition

A crawler is an independent script which independently browses the World Wide Web. This process of browsing is called Web crawling or spidering. A number of recent studies have noted that a tremendous amount of content is present on the net. Many sites use spidering as a means of providing up to date data to search engines. Web crawlers are used to create a copy of all the visited pages for later processing by a search engine that indexes the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, for example checking links or validating HTML code. Also, web crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses.

Net is full of useful data. From which some of the data is easily accessible to us with the help of standard search engine. We call such type of data as surface web. But the data which is not a part of surface web is called as deep web. It is also known as invisible data because it is not visible to the user directly. To access such type of data we need some special type of details as the password, authentication of the site etc. Accessing of deep web is now a big challenge for the researchers because it contains very important data. Public information on the deep Web is currently 400 to 550 times larger than the commonly defined World Wide Web[15]. A crawler is a part of a search engine. To search information from the internet, user submits his query to the search engine, search engine then reply the user with different links to different pages. User follows these links to collect the required information. Typically a search engine works in such a manner that special crawler software visits a site and reads the source code of the pages. This process is called as "crawling" or "spidering". Crawling starts with a seed URL and then follows the link presented on page using Depth First Search or Breadth First Search [1]. Then, this page is compressed and push into the search engine's repository called an "index". This stage is called as "indexing". Finally, when someone submits a query to the search engine, it retrieves that page out of the index and allocates it a certain rank. This process is called "ranking". Search engine can be categorized in three categories. Crawler based search engines use automated software programs called as ‘spiders’, ‘crawlers’, ‘robots’ or ‘bots’ to survey the web pages. Human-powered search engines believe on humans to submit information that is subsequently indexed and catalogued. Only information that is submitted by a human is put into the index. A search engine which works as a combination of both the type of search engine is called as hybrid search engine. Several major problems affect non-cooperative web crawlers on the web. The first problem is that web crawlers do not maintain a high-degree of freshness. The second is that multiple crawlers can redundantly crawls the same regions of the web. The third is that with the proliferation of web crawlers comes increased contention for shared network resources. This paper presented the technique which reduces the redundant data on net.

1.2 RELATED WORK

Web crawlers have always been a fascinating area of interest since the advent of web. A lot of work has also been done already to optimize the performances of web crawlers. They have certain Issues described below:

Crawler Architecture: Work done describes various architectures under which crawlers of certain current search engines are working. [2] Describes the architecture of the crawling technique used by Google whereas [3] describes a strategy based on text and link characteristics of referring pages by producing a rank function which is weighted sum of the text and link scores.[4] tells about a technique of effective focused crawling to improve the quality of web navigations. [5] Describes a parallel crawler with multiple architectures along with metrics for evaluation. The immense growing dimension of the World Wide Web induces many obstacles for all-purpose single-process crawlers including the presence of some incorrect answers among search results and the scaling drawbacks. As a result, more enhanced heuristics are needed to provide more accurate search outcomes in an appropriate timely manner. [6] This paper proposes the application of a link independent Web page importance metric to govern the priority rule within the crawl frontier through proposing a modest weighted architecture for a focused structured parallel Web crawler (CFP crawler) in which the credit assignment to URLs in crawl frontier is done according to a click stream-based prioritizing algorithm. A traditional crawler picks up a URL, retrieves the corresponding page and extracts various links, adding them to the queue. A deep Web crawler, after adding links to the queue, checks for forms. If forms are present, it processes them and retrieves the required information. [7] Analyze and compare important deep Web information crawling techniques to find their relative limitations and advantages. To minimize limitations of existing deep Web crawlers, a novel architecture is described based on QIIIEP specifications. The proposed architecture is cost effective and has features of privatized search and general search for deep Web data hidden behind html forms. The World Wide Web is an interlinked collection of billions of documents formatted using HTML. Due to the growing and dynamic nature of the web, it has become a challenge to traverse all URLs in the web documents and handle these URLs, so it has become imperative to parallelize a crawling process. The crawler process is further being parallelized in the form ecology of

Crawler workers that parallel download information from the web. [8] Proposes a novel architecture of parallel crawler, which is based on domain specific crawling, makes crawling task more effective, scalable and load-sharing among the different crawlers which parallel download web pages related to different domains specific URLs.

Page Update Policies: Each crawler has to update the web pages on a periodic basis to improve the quality of its databases. [9] Discusses scheduling algorithms for crawlers to index the web on a regular basis. [10] Describes the various freshness metrics used for gauging the freshness and quality of a local copy of a web page.

[16] Proposes various refresh policies and studies their effectiveness. The author first formalize the notion of freshness" of copied data by defining two freshness metrics, and then propose a Poisson process as the change model of data sources. Based on this framework, the author examines the effectiveness of the proposed refresh policies analytically and experimentally. [17] In this paper a hybrid approach is build on the basis of which a web crawler maintains the retrieved pages "fresh" in the local collection. Towards this goal the concepts of Page rank and Age of a web page is used. As higher page rank means that more number of users is visiting that very web page and that page has higher link popularity. Age of web page is a measure that indicates how outdated the local copy is. Using these two parameters a hybrid approach is proposed that can identify important pages at the early stage of a crawl, and the crawler re-visit these important pages with higher priority.

Page Priority Method: To prioritize a page over another, certain methods and parameters have been proposed that are also used by modern day search engines [11-13]. These methods take into account various parameters such as link count for a certain page or the keyword occurrence frequency to provide a suitable parameter for page relevance.

[14] Shows an algorithm to detect the changes in a web page. It shows the technique to update a page present in web repository after comparing it with the new copy.

2. EXISTING METHOD

Existing method consists of a multi threaded server and client crawlers. Server is used to store the data & to give the instruction to the client crawlers. Client crawlers are used to accept the instruction from the server & made its searches according to it. The server itself never downloads a page it just forward its request to one of its client to download the page. Once the page is downloaded it is stored in the web repository. A lot of the content on the web is dynamic. It is quite possible that after downloading a particular web page, the local copy of the page residing in the repository of the web pages becomes obsolete compared to the copy on the web. Therefore a need arises to update the database of web pages. The author discusses two methods [14] to derive certain parameters, which can help in deriving the fact whether the page has changed, or not. These parameters will be calculated at the time of page parsing. When the client counters the same URL again, it just calculates the code by parsing the page without downloading the page and compares it to the current parameters.

If changes in parameters are detected, it is concluded that the page has changed and needs to be downloaded again. Otherwise the URL is discarded immediately without further processing.

Changes in a web page is said to be occurred if there is a change in page structure or in content of the page. Pages on web are created using HTML or XML which uses tags. First method uses these tags to compare the web pages and second method creates a document tree.

3. PROPOSED METHOD

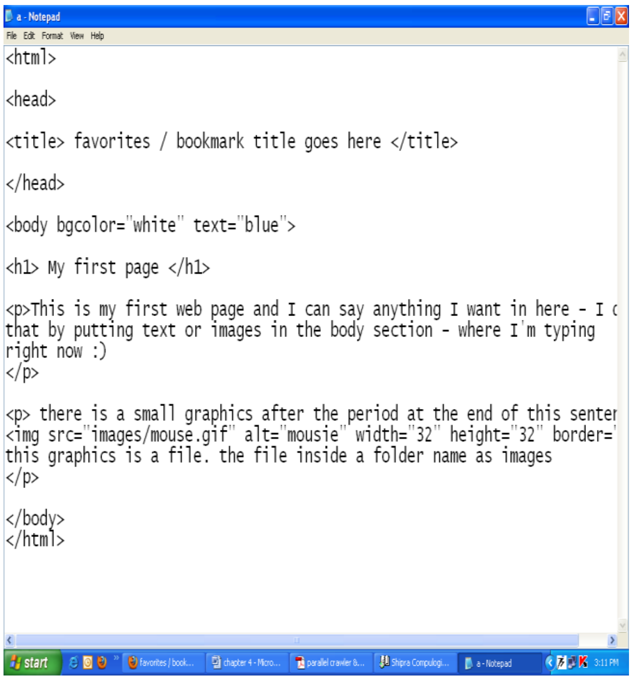
Once a decision has been taken to update the pages, it should be ensured that minimal resources are used in the process. Updating only those elements of the database, which have actually undergone a change, can do this. It also checks whether the page is already there in the database. If page is already present then we need to discard the link otherwise we replace the exiting page with the fresh downloaded page. In this paper we propose two methods to compare the structure of web page.

3.1 Method I

By structure we mean how the text images are displayed on the page. All of these objects are designed using HTML, XML or other formatting tools. All these formatting tools used tags arranging in proper manner. Changes in structure lead to rearrangement of these tags.

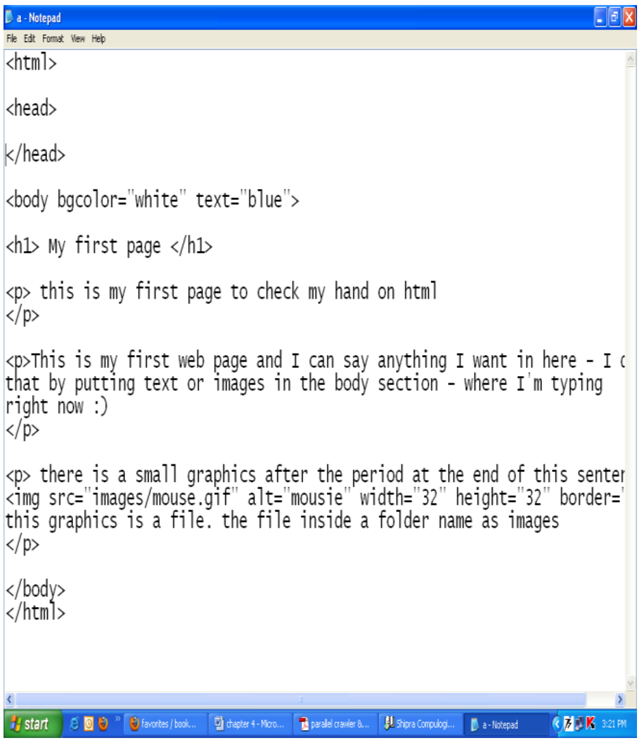
In this algorithm we start our process by comparing the first tag of existing page with the first tag of the new downloaded page. If we found it same the process is continued till end tag of the page otherwise page is declared to be changed from existing one.

For example consider the following page:



```
<html>

<head>

<title> favorites / bookmark title goes here </title>

</head>

<body bgcolor="white" text="blue">

<h1> My first page </h1>

<p>This is my first web page and I can say anything I want in here - I c
that by putting text or images in the body section - where I'm typing
right now :)
</p>

<p> there is a small graphics after the period at the end of this senter
<img src="images/mouse.gif" alt="mousie" width="32" height="32" border='
this graphics is a file. the file inside a folder name as images
</p>

</body>
</html>
```

Fig 1.1: Web page in web repository



```
<html>

<head>

</head>

<body bgcolor="white" text="blue">

<h1> My first page </h1>

<p> this is my first page to check my hand on html
</p>

<p>This is my first web page and I can say anything I want in here - I c
that by putting text or images in the body section - where I'm typing
right now :)
</p>

<p> there is a small graphics after the period at the end of this senter
<img src="images/mouse.gif" alt="mousie" width="32" height="32" border='
this graphics is a file. the file inside a folder name as images
</p>

</body>
</html>
```

Fig 1.2: Web page on fresh link

From the above two pages (fig 1.1 and fig 1.2) following table is formed:

Tag Name	**Web Page in Web Repository**	**Web Page on Fresh Link**
Tag1	H	H
Tag2	H	H
Tag3	T	B

Table: 1.1 (Comparison of tags of web page fig 1.1 and fig 1.2)

At Tag1 both tags are same so the process is continued for next tag. Next tag is also same so the process moves to next tag. This tag is different. So we declared that the page has changed.

The proposed method offers following advantages:

- We are using just a single tag rather than using whole string. It reduces the inconvenience occurred in creating a whole string by reading the complete web page.
- Rather than comparing the complete string we will compare it till the third tag which is a changed value. So it will reduce the comparison time.
- It identifies even the single tag changed in the structure.
- Even a single change in the structure of a page can be tracked.
- Most of the time we are able to calculate the actual result by comparing just one tag.
- It works for all type of formatting styles.
- To take the decision page downloading the client crawler only needs to compare these tags.

3.2 Method Two

It includes two methods to detect the changes on a web page: construction of document tree and level order tree traversing.

The structure of every node used in tree presentation of web page should contain following information.

ID: it stores the unique id for each node of the tree.

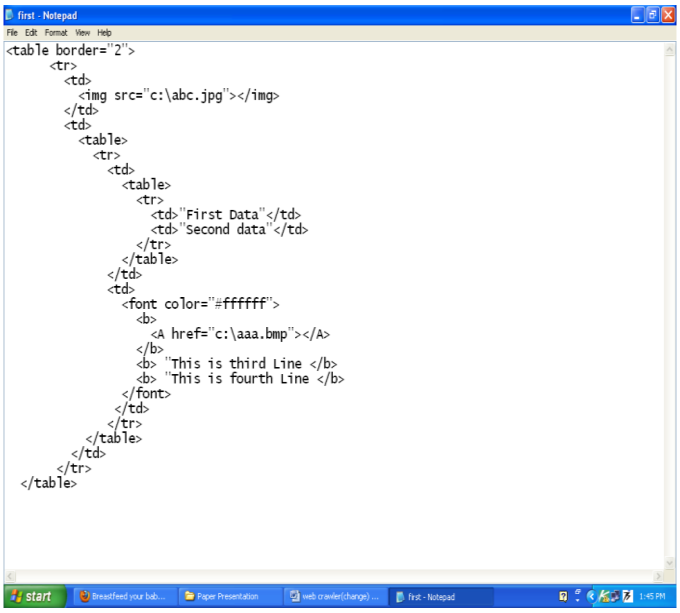
CHILD: it stores the information about children of each node.

LEVEL: it stores the information about the level of a node.

LEVEL_ARRAY: it is the array which stores the level of each node.

For example we get tree structure shown in fig 1.5 from web page shown in Fig. 1.3 and we get tree structure as is shown in Fig. 1.6 from the web page shown in fig 1.4.

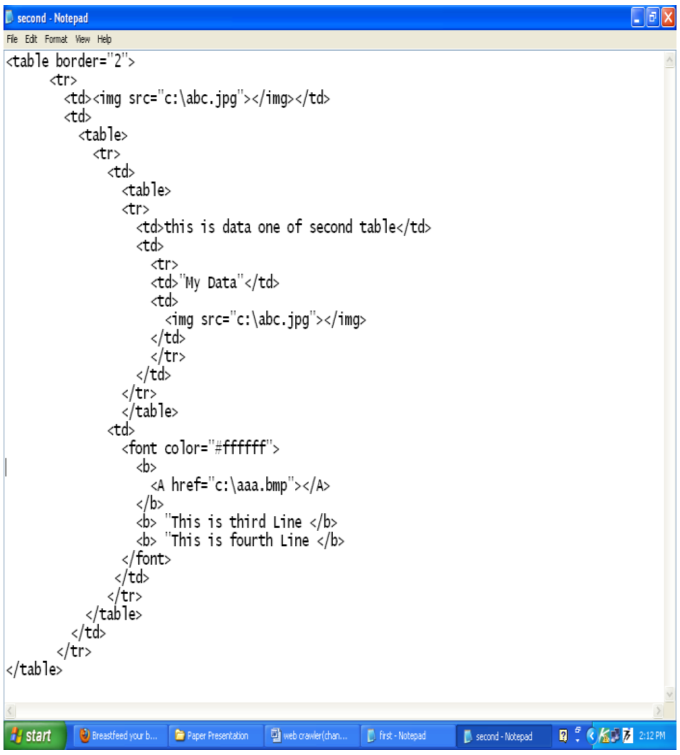
Here in this example we are considering fig 1.3 as the initial web page and fig 1.4 as the changed web page. With the help of document tree created in fig 1.5 and 1.6 following table 1.2 is drawn. We are using BFS to create the table.



```
<table border="2">
      <tr>
        <td>
          <img src="c:\abc.jpg"></img>
        </td>
        <td>
          <table>
            <tr>
              <td>
                <table>
                  <tr>
                    <td>"First Data"</td>
                    <td>"Second data"</td>
                  </tr>
                </table>
              </td>
              <td>
                <font color="#ffffff">
                  <b>
                    <A href="c:\aaa.bmp"></A>
                  </b>
                  <b> "This is third Line </b>
                  <b> "This is fourth Line </b>
                </font>
               </td>
              </tr>
           </table>
         </td>
       </tr>
  </table>
```

Fig 1.3: Web repository page

For Example



```
<table border="2">
      <tr>
        <td><img src="c:\abc.jpg"></img></td>
        <td>
          <table>
            <tr>
              <td>
                <table>
                <tr>
                  <td>this is data one of second table</td>
                  <td>
                    <tr>
                    <td>"My Data"</td>
                    <td>
                      <img src="c:\abc.jpg"></img>
                    </td>
                    </tr>
                  </td>
                </tr>
                </table>
              <td>
                <font color="#ffffff">
                  <b>
                    <A href="c:\aaa.bmp"></A>
                  </b>
                  <b> "This is third Line </b>
                  <b> "This is fourth Line </b>
                </font>
               </td>
              </tr>
           </table>
         </td>
       </tr>
</table>
```

Fig 1.4: Fresh link page

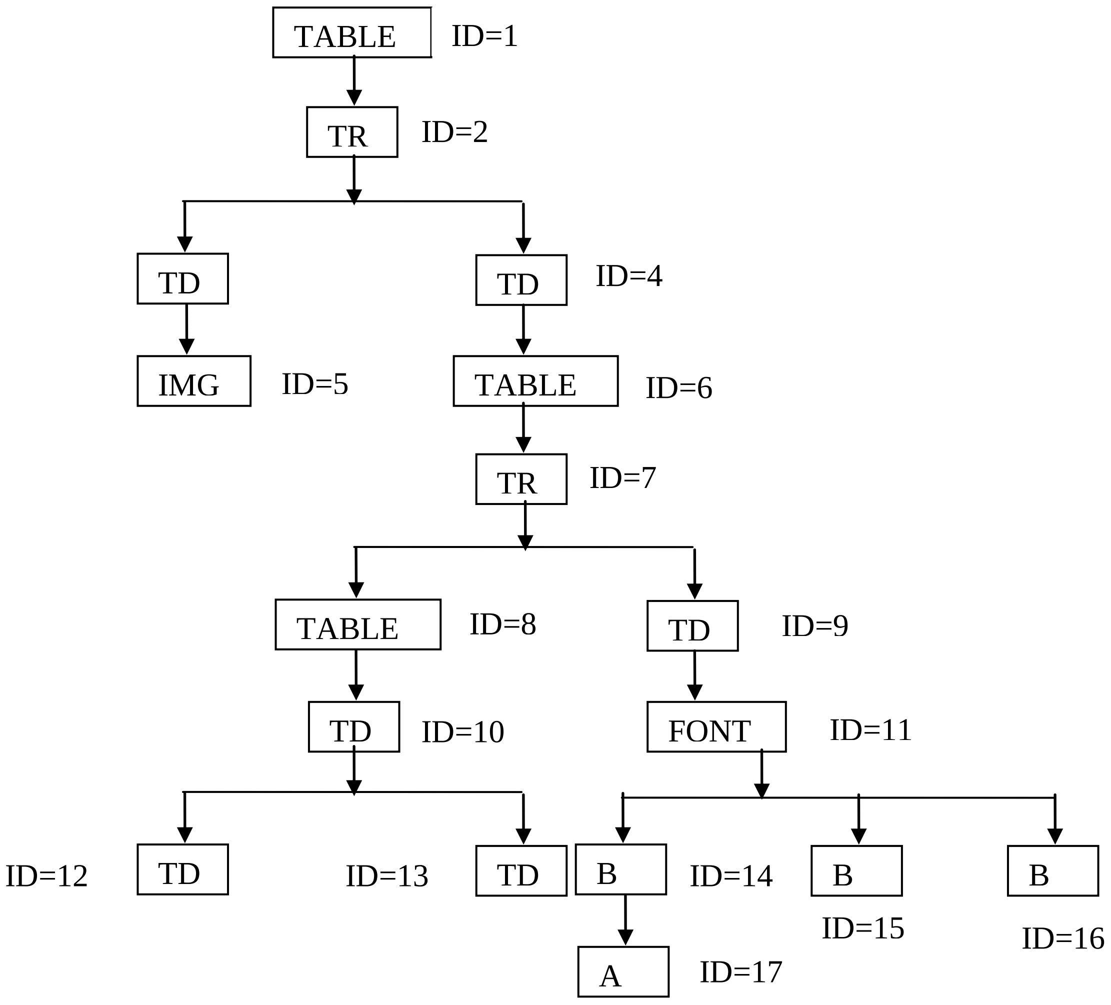


Fig 1.5: Tree structure from web repository page

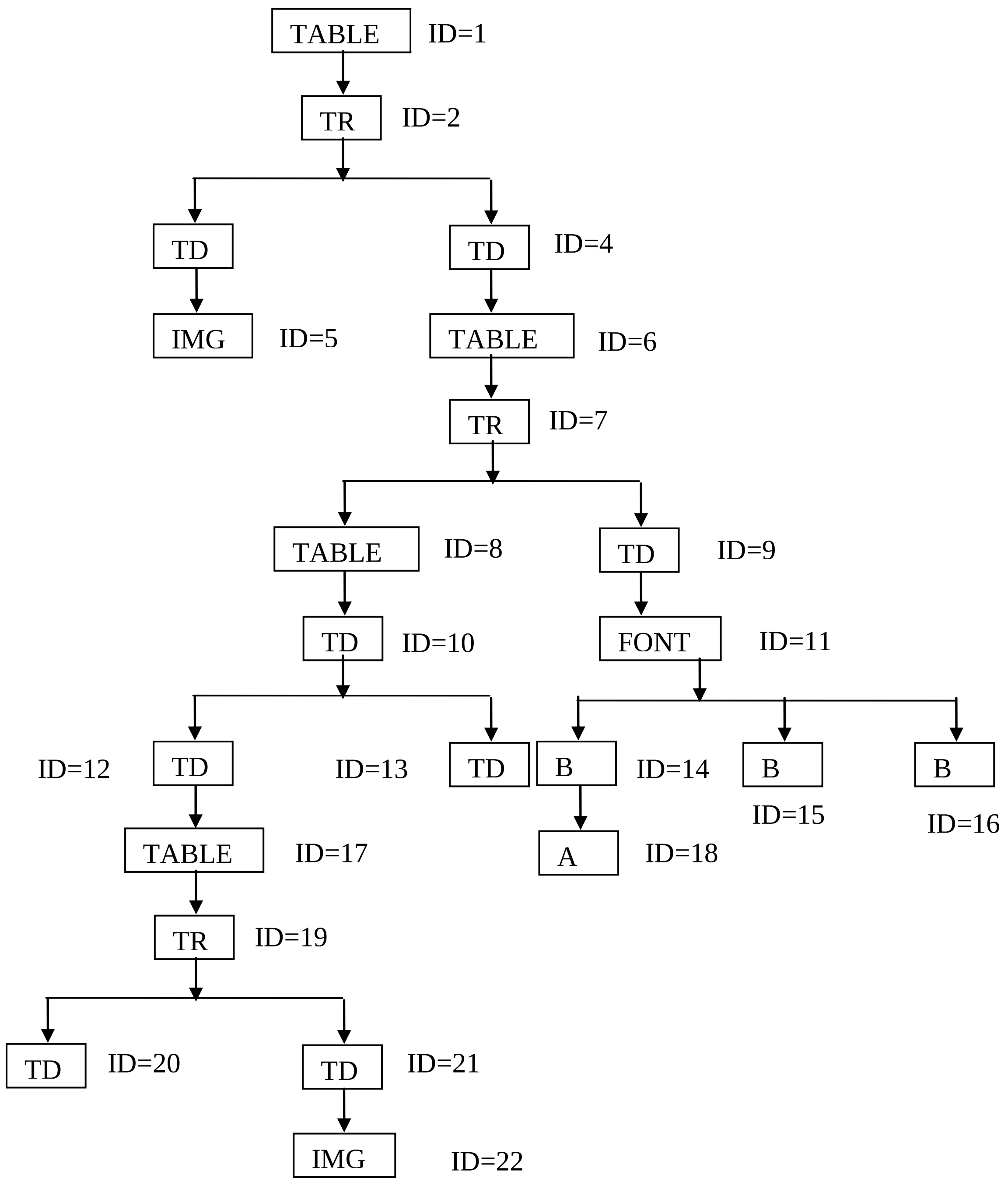


Fig 1.6: Tree structure from fresh link page

LEVEL	INITIAL STRUCTURE	MODIFIED STRUCTURE
Level -1	1	1
Level -2	2	2
Level -3	2	2
Level -4	2	2
Level -5	2	2
Level -6	1	2
Level -7	2	2
Level -8	5	5
Level -9	**1**	**2**

Table 1.2: Level Structure using BFS

According to initial structure

LEVEL = {1, 2, 3, 3, 4, 4, 5, 6, 6, 7, 7, 8, 8, 8, 8, 8, 9}

ID= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17}

CHILD= {1, 2, 1, 1, NULL, 1, 2, 1, 1, 2, 3, NULL, NULL, 1, NULL, NULL, NULL}

LEVEL_ARRAY = {1, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 8, 8, 8, 9}

According to modified structure

LEVEL = {1, 2, 3, 3, 4, 4, 5, 6, 6, 7, 7, 8, 8, 8, 8, 8, 9, **9**}

ID= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, **17, 18**}

CHILD= {1, 2, 1, 1, NULL, 1, 2, 1, 1, 2, 3, NULL, NULL, 1, NULL, NULL, **1, NULL**}

LEVEL_ARRAY = {1, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 8, 8, 8, 9, 9, 10, 11, 11, 12}

Now we start level wise traversing of both the tree. Results are shown in the table. As per the example after comparing both the set we get the idea that structure has changes at level-9. As soon as we find the first difference among these two structures a conclusion has been drawn that page has changed and we need to update the web repository. This approach gives us following benefits:

- Rather than comparing the whole document we make our search till the first difference.
- Even if a node is deleted or added, it is easily recorded by the LEVEL_ARRAY.
- The client crawler needs to check only two sets of data at the time of updating.

4. CONCLUSION & FUTURE SCOPE

With the size of the web increasing at a tremendous rate, web crawlers are being more and more challenged to discover and maintain web objects, especially on behalf of web search engines. Currently, web search engines relying on web crawlers to keep their indices up-to-date are falling behind. Web page change detection is very important as it provide us with the knowledge that what is happening on the Web. It keeps our data updated. Certain techniques can ensure that the most popular material is kept up-to-date, but that is limiting. In this paper we proposed a general protocol to detect the changes that occur in the web pages presented on the internet. It consists of two methods. Firstly we try to detect the changes in page structure by comparing the tags in both the paper. In second method we create a document tree. Then we do level wise traversing to track the changes among two pages. Finally we declare the result whether the web pages are same or not.

Our future work consist of more research to minimize the efforts to detect the changes occurred in web. My research on the web page change detection is not finished since I am trying find other ways to compare two pages which cost minimum in terms of memory, bandwidth etc.

5. REFERENCES

[1] David Eichmann, “The RBSE Spider – Balancing effective search against web load”, Repository Based Software Engineering Program , Research Institute for Computing and Information Systems, University of Houston – Clear Lake.

[2] Sergey Brin and Lawrence Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, In Proceedings of the Seventh World-Wide Web Conference, 1998.

[3] Anshika pal, Deepak Singh tomar, S.C srivastava, “effective focused crawling based on content and link structure analysis”, international journal of computer science and information security, vol 2, no. 2, June 2009

[4]jody Johnson, kostas Tsioutsiouliklis, C.L Giles, “Evolving strategies for focused web crawling”, Proceedings of twentieth international conference of machine learning, Washington DC, 2003.

[5] Junghoo Cho & Hector Garcia-Molina, “Parallel Crawlers”. Proceedings of the 11th international conference on World Wide Web WWW '02, Honolulu, Hawaii, USA. ACM Press. Page(s): 124 – 135.

[6] F. Ahmadi Abkenari, Ali Selamat, “A clickstream based focused trend parallel web crawler”, vol 9, no 5, November 2010.

[7] Dilip Kumar Sharma, A. K. Sharma,” A Novel Architecture for Deep Web Crawler”, International Journal of Information Technology and Web Engineering, vol 6, issue 1, 25-48, January-March 2011

[8] Nidhi Tyagi, Deepti Gupta, “A novel architecture for domain specific parallel crawler”, Indian journal of computer science and engineering, vol 1, no 1, 44 – 53.

[9] E. Co.man, Jr., Z. Liu, and R. R. Weber, “Optimal robot scheduling for web search engines”. Proceedings of the 11th international conference on World Wide Web WWW '02 Honolulu, Hawaii, USA. ACM Press. Page(s): 136 – 147.

[10] “Synchronizing a database to improve freshness, submitted for publication”. Proceedings of the 2000 ACM SIGMOD international conference on Management of data. Volume 29 Issue 2. Page(s): 117 – 128.

[11] M. Diligenti, F. M. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, “Focused crawling using context graphs”, In Proceedings of the Twenty-sixth International Conference on Very Large Databases, 2000.

[12] S. Chakrabarti, M. van den Berg, and B. Dom, “Focused crawling: A new approach to topic-specific web resource discovery”, In The 8th International World Wide Web Conference, 1999.

[13] Junghoo Cho, Hector Garcia-Molina, and Lawrence, “Efficient crawling through URL ordering Page”, In Proceedings of the 7th World-Wide Web Conference, 1998, page(s):161-171.

[14] Divakar Yadav, A.K Sharma, J.P. Gupta, “ Parallel crawler architecture and web page change detection”, WSEAS transaction on computers, issue 7, volume 7, july 2008

[15] Bergman, Michael K, “White paper: the deep web : surfacing hidden value”, Vol 7, Issue 1, August 2001

[16] Junghoo Cho , Hector Garcia-molina ,” Effective page refresh policies for web crawlers”,Vol 28, Issue 4, December 2003, Pages 390 – 426

[17] Vipul Sharma, Mukesh Kumar, Renu Vig, A Hybrid Revisit Policy For Web Search, Vol 3, No 1, Feb 2012, Page(s): 36 - 47