I.J. Modern Education and Computer Science, **2015, 9, 36-43**
Published Online September 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.09.05

A Novel Web Page Change Detection Approach using Sql Server

Md. Abu Kausar
Dept. of Computer & System Sciences, Jaipur National University, Jaipur, India
Email: kausar4u@gmail.com

V. S. Dhaka
Dept. of Computer & System Sciences, Jaipur National University, Jaipur, India
Email: vijaypal.dhaka@gmail.com

Sanjeev Kumar Singh
Dept. of Mathematics, Galgotias University, Gr. Noida, India
Email: sksingh8@gmail.com

***Abstract*—**The WWW is very dynamic in nature and is basically used for the exchange of information or data, the content of web page is added, changed or deleted continuously, which makes Web crawlers very challenging to keep refresh the page with the current version. The web page is frequently changes its content hence it becomes essential to develop an effective system which could detect these types of changes efficiently in the lowest browsing time to achieve this changes. In this chapter we compare the old web page hash value with the new web page hash value if hash value is changed that means web page content changed. The changes in web page can be detected by calculating the hash value which is unique.

***Index Terms*—**Web Page change detection, Hash Value, Checksum, HTML.

I. Introduction

Nowadays people use internet for exchange of information across the world which results in uploading of new information on web sites very frequently. Web sites bring new and latest information in front of the user as a web page. The web is updated very frequently with new web pages that contain more relevant information to the user. Web page changes will be categorized as Structural Change i.e., addition or removal of tags, content changes. i.e. change of main content of web page, cosmetic changes e.g., changes in the font, color etc., Behavioral Change i.e., changes of script, applet etc. A client may visit a particular web site frequently and is concerned to see how every web page has changed from its previous version. For instance, a cricket lover will be concerned to know how the score is changing continually at every minute. Likewise, to locate any specific things in the web pages which change much of the time for instance one could be keen on knowing the positioning of the hit films. Here changes in the positioning relate to changes in the web page layout. Therefore this method supports to decrease the searching time of the client and permits the client to discover the things in web documents which changes regularly. Client may be involved in knowing the positioning of the hit movies. Here changes in the positioning of the movies match with changes in the design of the web page. Hence this framework assistance to decrease the searching time of the client and lets the client to locate the matters in web documents which changes regularly. Web pages content changes at very fast rate. Thus it becomes very difficult to examine the changes which are made in web pages and retrieve the original web pages at fast rate. About 60% of the contents on the Web is dynamic [1] and approx 14% of the links in search engines are broken [2]. Search engines depend on web crawlers to collect web pages and store the web pages in their local repository, and the copies of web pages are later saved to reply the related questions of user. The search engines may match these retrieved copies with their available user queries, due to limitations of resources it is not possible for web crawlers to continually observe and retrieve all web pages.

This paper describes a fast HTML web page change detection approach that saves computation time among two versions of a web page by calculating hash value of both web pages. Hash value is calculated & store in database in case of web page change detection the stored hash value is compared with current hash value of web page. If hash value is different then replace the old value with new one.

The rest of this paper is organized as follows: The existing work related to our problem is presented in Section 2. Section 3 describes the classification of web page change Section 4 describes the architecture of proposed system and Section 5 concludes this paper.

II. Related Work

To predict the probability of web page change detection is not a new problem and there are so many research papers [3], [4], [5], [6], [7], [8], [9]. For example, Cho and Garcia-Molina [3] crawled approx 720,000 web documents in a day for a period of 4 months and examine how the web page changes. Ntoulas, Cho, and Olston [8] collect snapshots of approx 150 websites weekly for one year and examine web page change. The author found that most web documents did not change as per packs of words measure of comparability. Even for web documents which are changed, the change amount is minor. Recurrence of web page change was not a major indicator of the measure of web page change; however the measure of web page change was a decent indicator of the prospect measure of web page change.

The biggest examine of web page change detection was carry out by Fetterly et al. [4]. The author crawled approx. 150 million web documents in a week for eleven weeks, and analyzed the changes of web documents. Ntoulas et al. [8], they discovered a moderately little measure of page change, by approx 65% of web document sets remain closely the similar. The study also establish that previous web page change was a decent predictor of prospect change, that web document measurement was connected with age change, and the top level area of a web document was associated with change.

Olston and Panday [10] crawled approx 10,000 random Web pages and approx 10,000 web pages tested from the Open Directory for a number of months in every two days. Their examinations measured both change recurrence and data life span, and found only a reasonable association between the two.

[11] and [12] discussed how the retrieved outcomes change over time. The main aim of these studies about the understanding of the changing aspects of searching tools and the concerns modify for explore who need to revisit to formerly observed web documents.

In [13], E. Leonardi and Bhowmick studied that web page change detection using ordered and unordered tree model method is not appropriate for the reason that many memories needed for keeping two different types of XML records in the memory. In this research work author use relational database for identifying content variations of ordered large XML document. Firstly it saves XML data in relational database and then identifies the variations of web pages by the use of SQL queries. This method provides the improved scalability and enhanced execution.

XMLTreeDiff [14] figures variance among XML Documents taking into account heuristic closeness. It processes the checksum values for the hubs of both the records utilizing DOM checksum [15] and after that decreases the measure of the two trees by evacuating indistinguishable sub trees.

In [16] [17] [18], different diff algorithms are clarified for identifying modifications in XML data. The algorithm in [18] is taking into account discovering and afterward taking out the similar hubs from the two XML documents that are being analyzed. The change operations are next detected from the non-matching nodes. Coordinating of hubs is taking into account looking at elements of hub substance and children (signatures) and request of event in common manner request subsequences of hubs. In [16] [17] utilize alter scripting to analyze between two web pages and convert the web documents to trees as per the XML configuration. The power of these algorithm lies in their low time-complexity, which is in the order of O(nlogn). Though, this great execution can't be attained when looking at HTML pages as it depend on certain XML structures. Alter scripting only is not enough for attaining O(nlogn), particularly if move operations are to be considered.

In [19] convert a HTML page into a tree structure and classify node data into attributes, content and configuration. Three likeness methods are utilized to identify variations of these three types, intersect i.e., rate of words which are similar, attdist i.e., amount of the relative weight of similar attributes, and typedist i.e. measure of the location of the elements in the tree. The system hypothetically uses the Hungarian algorithm [20] for searching the most similar subtree between two web pages, but no such type of details on its uses are given. The experimental results do not discuss speed performance they only display the effects of the importance actions on finding correctness. In [21], a novel web page change detection method for static web pages was proposed which is taking into account into three heuristic-derived enhancements to the Hungarian algorithm. For the Hungarian algorithm the comparability coefficients among all hubs of each subtree in the mention page and those of each subtree in the upgraded page are calculated and set in a cost matrix. The primary advancement limits the utilization of the calculation to the subtrees having the conceivable of being the most related subtrees, the second manages the halting criteria of the calculation if the span of the upgraded subtrees page gets to be not as much as that of the subtree of the notice page by a particular issue, lastly, the third improvement stops the calculation after the similarity coefficients drop beneath a certain edge.

III. Web Page Change Classification

The changes of web pages can be classified into following four main categories [22]:

A. Structural Change
B. Content Change
C. Cosmetic Change
D. Behavioral Change

A. Structural Change

The structure of web page is change by addition or deletion of some tags. The addition or deletions of link in a web page also change the whole structure of the web page.

```
<html>
<head>
<title>Welcome to my Web page</title></head>
<body>
<p>----------------------------------------------</p>
<b>----------------------------------------------</b>
<p font="--" color="-------">-------------------<p>
<marquee>------------------------------------</marquee>


</body>
</html>
```

Fig.1. Initial Version of Web Page.

```
<html>
<head>
<title>Welcome to my Web page</title></head>
<body>
<p>----------------------------------------------</p>
<b>----------------------------------------------</b>
<p>-----------------------------------------------<p>


</body>
</html>
```

Fig.2. Changed Version of Web Page.

B. *Content Change*

When the main content of the web page is change then this type of change comes under content change. For example, web page created for online news site change very frequently to keep latest information from time to time. Another example can take of web page created for any match which is continuously updated as the tournament progress. When the match ends, the web page displays the information about award ceremony instead of match scores. The web page which contains the records of the players get modified according to reflect the latest records.

```
<html>
<head> <title> Welcome to my website</title>
</head>
<body><center>Times of India Sports 1-March-2014
</center>
<ul>
<li> Afghanistan set 255-run target for B'desh
<li> Suicide at metro station in Kolkata, services
affected
<li> Asia Cup: India need to make amends against Pak
<li> Asia Cup: Sri Lanka beat India by two wickets.
</ul>
</body>
</html>
```

Fig.3. Initial Version of Content Change of Web Page.

```
<html>
 <head> <title> Welcome to my website</title>
</head>
<body><center>Times of India Sports 1-March-2014
</center>
<ul>
<li> Asia Cup 2014: Stanikzai, Shenwari lead
Afghanistan to 254/6 against Bangladesh
<li> Suicide at metro station in Kolkata, services
affected
<li> President rule in Andhra Pradesh, assent to
Telangana bill.
<li> Asia Cup: Sri Lanka beat India by two wickets.
</ul>

</body>

</html>
```

Fig.4. Changed Version of Content Change of Web Page.

C. *Cosmetic Change*

When the HTML tag of the web page is changed without changing the content of the web page then this type of change comes under cosmetic change. The content of web page is remained.

```
<html>
 <head> <title> Welcome to my website</title> </head>
<body><center>Times of India Sports 1-March-2014
</center>
<p style="background-color: red"><b> Afghanistan set
255-run target for B'desh</b></p>
<p> Suicide at metro station in Kolkata, services
affected</p>
<p style="background-color: yellow"><u> Asia Cup:
India need to make amends against Pak</u></p>
<p style="background-color: green"><u> Asia Cup: Sri
Lanka beat India by two wickets.</u>

</p>

</body>

</html>
```

Fig.5. Initial Version of Cosmetic Change of Web Page.

D. *Behavioral Change*

When scripts, applets etc. in web page gets modified then this type of modifications comes under behavioral change. This type of codes (Script, applets etc.) is especially hidden in other files. However, it is difficult to grasp such modifications.

```
<html>
 <head> <title> Welcome to my website</title>
</head>
<body><center>Times of India Sports 1-March-2014
</center>
<p> Afghanistan set 255-run target for B'desh</p>
<p> Suicide at metro station in Kolkata, services
affected</p>
<p> Asia Cup: India need to make amends against
Pak</p>
<p> Asia Cup: Sri Lanka beat India by two wickets.
</p>

</body>

</html>
```

Fig.6. Changed Version of Cosmetic Change of Web Page

IV. Extraction of Web Page Changes Between Different Versions

HTML pages are made by markup languages which can be used to construct a document tree. For instance, the Document Object Model [23] builds a document tree for a HTML pages. The document tree of web pages brings more data than a content document and extracts more information efficiently than a text file.

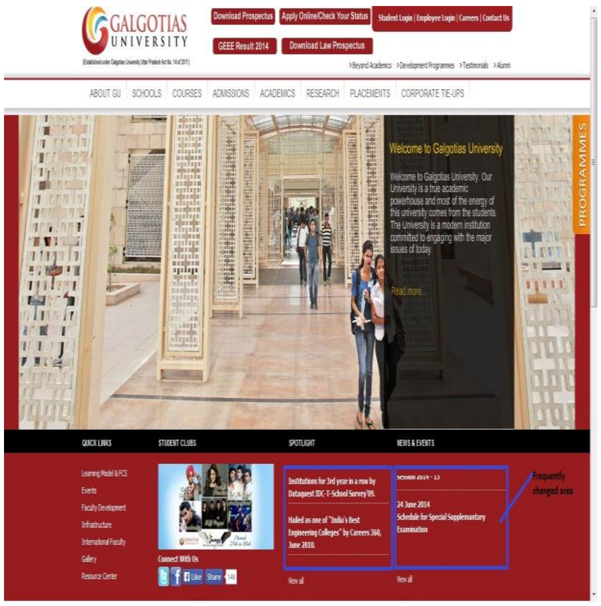


Fig.7. An Example of Modification Areas on Web Pages.

On average the quantity of change on current web document is considerably lesser than the content of the web page. At the point when users add or change the content of page, it is possible that only content is included or changed whereas the web page tree structure and design of content in web pages remain comparatively fixed. Nowadays more web developers have a tendency to keep up the structure of site pages as layouts, they modify the web pages in some specific portions like latest news and events. Figure 7 is an example of web page using design of templates. The blocks surrounded by the blue boxes are generally used to display fresh information on this web page.

To accommodate modification on web pages, we use document tree to denote different types of web pages in takeout web changes. Our algorithm is carried out in two different stages:

- Construction of document tree
- Calculate Hash value of web pages

A. Document tree construction

We use Document Object Model (DOM) [23] to construct HTML document tree. The components in HTML pages are not all the time viewed in a nested way. While constructing the HTML document tree following two difficulties are faced.

- According to HTML 4.01 Description [24], the presence of end tag can be compulsory or noncompulsory or not allowed. For instance, <B>, <div>, <i>, <table>, <a> are required tags, <p>, <li> are optional tags and <meta>, <br> are not allowed tags.
- Certain designing tags may not align properly, for example <td>. Figure 8 illustrates how misalign tag can be found. Some HTML parsers, for example, Microsoft Internet Explorer, can put up such misalign tag to show the web page properly.

```
<html>
<head>
<title> This is my site </title>
</head>
<body> Hello
<Table border=2>
<tr>
<td>Apple</td>
<td>Mango
</tr>
</td>                              /*misalign    tag*/
</Table></body></html>
```

Fig.8. Misalign of HTML Tags.

The process for constructing a HTML document tree is generally included in a HTML document parser. The parser break down the content of pages and returns the information when text data or start tag or end tag are available. The document tree constructors keeps up a pointer beginning at the root level, includes nodes and pass through the tree in the course of parsing. At the point when an end tag is absent, it includes an extra end tag for maintaining the nested structure. There is an exceptional situation where several optional tags appear in the same subtree without end tags for e.g. <ol> <li> ... <li> ... <li> ... </ol>). Instead of insertion them in a nested subtree, the parser inserts them at the same level in the <ol> tag sub-tree. Figure 9 demonstrates the change when many similar optional tags are in the same depth and leave the end tags. In the design of various same optional tags for example <li>, flat structure is preferred over nested structure.

```
<html>                          <html>
<head>                          <head>
<title> Hello </title>          <title> Hello </title>
</head>                         </head>
<body>                          <body>
<ol>                            <ol>
<li>Apple                       <li>Apple </li>
<li>Orange       ➡             <li>Orange </li>
<li>Grapes                      <li>Grapes </li>
<li>Mango                       <li>Mango </li>
</ol>                           </ol>
</body>                         </body>
</html>                         </html>
```

Fig.9. Tag Structure Transformations of Optional Tags.

B. *Calculate Hash value of web pages*

Change detection is refined by matching the recently retrieved web page with an already retrieved page saved on database. Change detection is taking into account computing the Hash value of two HTML documents. The web pages which are not 100% similar are considered as changed at server side.

The change detection method computes checksum for the entire web page. While updating the web page in the database, comparison between the checksums of both versions of web pages are performed and in case of any difference, it is fulfilled that the web page at web server site has been modified as compared to the local copy of web page maintained in the database at search engine end

V. Architecture Of The Proposed System

The operation of the proposed approach is shown in Figure 10.

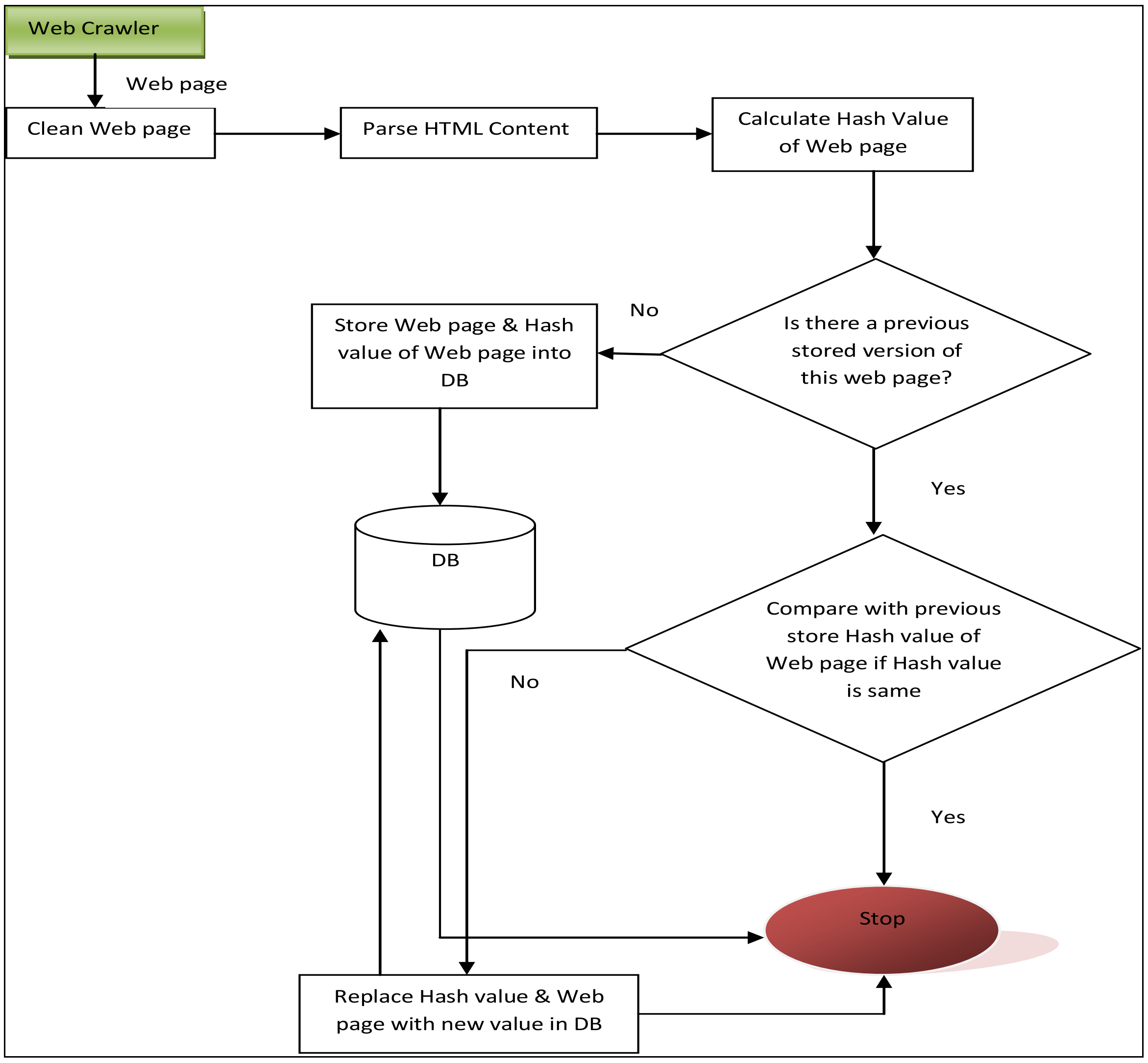

Web Crawler
Web page
Clean Web page
Parse HTML Content
Calculate Hash Value of Web page
Is there a previous stored version of this web page?
No
Store Web page & Hash value of Web page into DB
Yes
DB
Compare with previous store Hash value of Web page if Hash value is same
No
Yes
Stop
Replace Hash value & Web page with new value in DB

Fig.10. Diagram of Proposed Approach.

The whole process is started by the web crawler. The web crawler downloads the web pages from the web then it parses the HTML content from the web pages. After parsing the HTML content the hash value is calculated then this hash value is compared with previously stored hash value, if hash value is same as stored in database then it means web page is not updated with new value. If hash value is changed then it means web page changed its value, in this case old web page is updated with new web page and hash value is also updated and new value is stored in database. Every time this process runs, it checks to determine if there are web pages that need to be downloaded and compared to their equivalent previously stored versions.

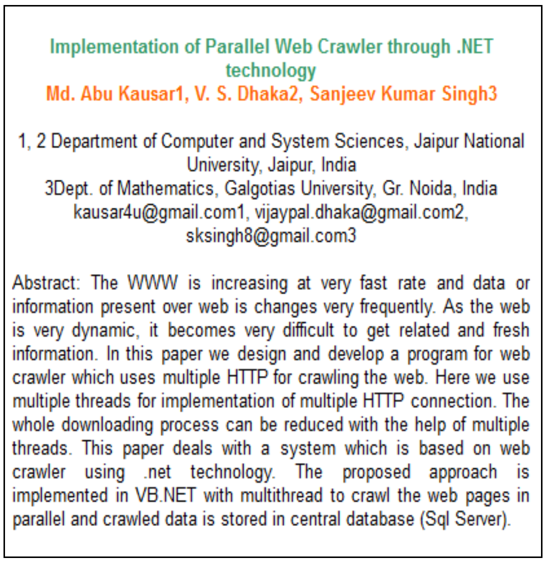
Change detection is refined by matching the recently retrieved web page with an already retrieved page saved on database. Change detection is taking into account computing the Hash value of two HTML documents. The web page portions which are not 100% similar are considered to have been changed. Figure 10 shows proposed approach.

The change detection method computes checksum for the entire web page. While updating the web page in the database, comparison between the checksums of both versions of web pages are performed and in case of any difference, it is fulfilled that the web page at web server site has been modified as compared to the local copy of web page maintained in the database at search engine end.

We use Sql Server inbuilt checksum() function for detecting any change in the web page whether the web page change is of any types like (Structural Change, Content Change, Cosmetic Change or Behavioral Change). Any type of change can be detected by calculating checksum value. Given below figure shows the previous and changed version of web page.

In Figure 12 Changed text is shown in red color.

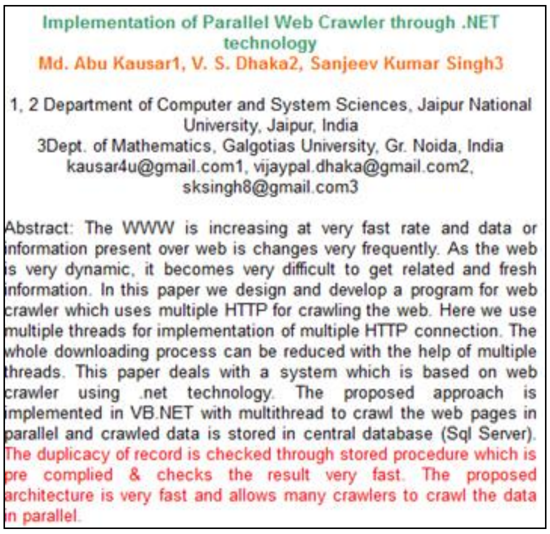
Figure 13 shows different checksum value of both web pages.


Implementation of Parallel Web Crawler through .NET technology
Md. Abu Kausar1, V. S. Dhaka2, Sanjeev Kumar Singh3

1, 2 Department of Computer and System Sciences, Jaipur National University, Jaipur, India
3Dept. of Mathematics, Galgotias University, Gr. Noida, India
kausar4u@gmail.com1, vijaypal.dhaka@gmail.com2, sksingh8@gmail.com3

Abstract: The WWW is increasing at very fast rate and data or information present over web is changes very frequently. As the web is very dynamic, it becomes very difficult to get related and fresh information. In this paper we design and develop a program for web crawler which uses multiple HTTP for crawling the web. Here we use multiple threads for implementation of multiple HTTP connection. The whole downloading process can be reduced with the help of multiple threads. This paper deals with a system which is based on web crawler using .net technology. The proposed approach is implemented in VB.NET with multithread to crawl the web pages in parallel and crawled data is stored in central database (Sql Server).

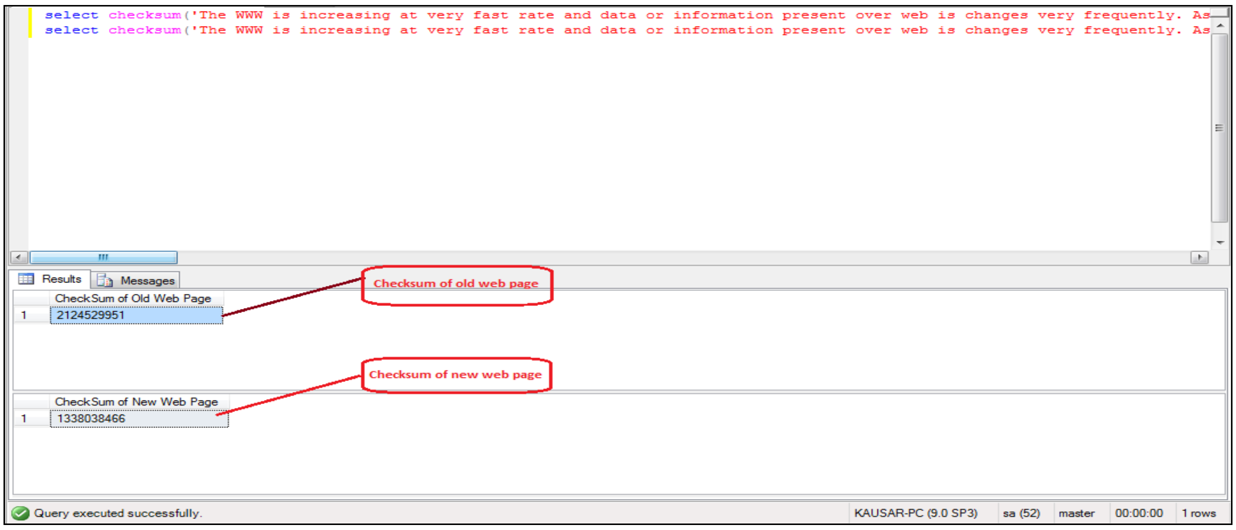
Fig.11. Initial Version of Web Page.


Implementation of Parallel Web Crawler through .NET technology
Md. Abu Kausar1, V. S. Dhaka2, Sanjeev Kumar Singh3

1, 2 Department of Computer and System Sciences, Jaipur National University, Jaipur, India
3Dept. of Mathematics, Galgotias University, Gr. Noida, India
kausar4u@gmail.com1, vijaypal.dhaka@gmail.com2, sksingh8@gmail.com3

Abstract: The WWW is increasing at very fast rate and data or information present over web is changes very frequently. As the web is very dynamic, it becomes very difficult to get related and fresh information. In this paper we design and develop a program for web crawler which uses multiple HTTP for crawling the web. Here we use multiple threads for implementation of multiple HTTP connection. The whole downloading process can be reduced with the help of multiple threads. This paper deals with a system which is based on web crawler using .net technology. The proposed approach is implemented in VB.NET with multithread to crawl the web pages in parallel and crawled data is stored in central database (Sql Server). The duplicacy of record is checked through stored procedure which is pre complied & checks the result very fast. The proposed architecture is very fast and allows many crawlers to crawl the data in parallel.

Fig.12. Changed Version of Web Page.


```
select checksum('The WWW is increasing at very fast rate and data or information present over web is changes very frequently. As
select checksum('The WWW is increasing at very fast rate and data or information present over web is changes very frequently. As
```

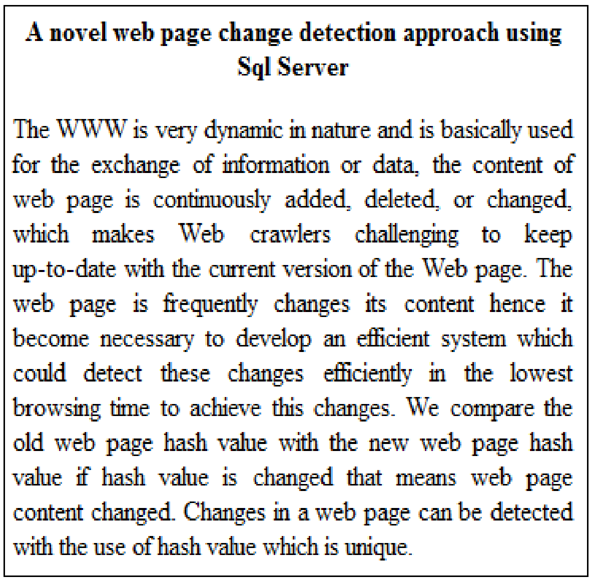
```
   CheckSum of Old Web Page
1  2124529951

   CheckSum of New Web Page
1  1338038466
```

Query executed successfully. KAUSAR-PC (9.0 SP3) sa (52) master 00:00:00 1 rows

Fig.13. Checksum Values of Both Web Pages.

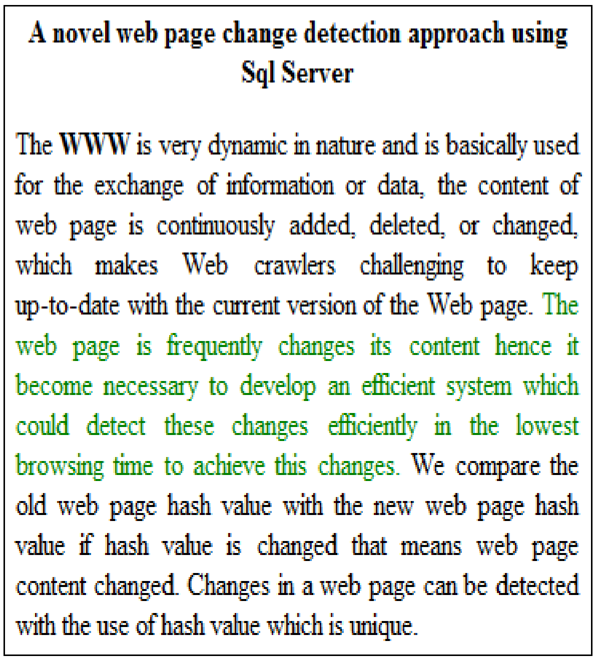
So we can say that content of web page is changed. We use Sql Server for evaluate checksum of both web pages which give the result very fast and takes very less time for evaluate checksum.

The proposed approach can also detect cosmetic change. If a web page change its font color or it changes its font weight then we can evaluate both web page checksum value. There are two web pages shown in Figure 14 and 15. Figure 14 shows simple web page, in Figure 15 the font color is changes as green and some content is in bold. Fig. 16 shows checksum value of both web pages.


A novel web page change detection approach using Sql Server

The WWW is very dynamic in nature and is basically used for the exchange of information or data, the content of web page is continuously added, deleted, or changed, which makes Web crawlers challenging to keep up-to-date with the current version of the Web page. The web page is frequently changes its content hence it become necessary to develop an efficient system which could detect these changes efficiently in the lowest browsing time to achieve this changes. We compare the old web page hash value with the new web page hash value if hash value is changed that means web page content changed. Changes in a web page can be detected with the use of hash value which is unique.

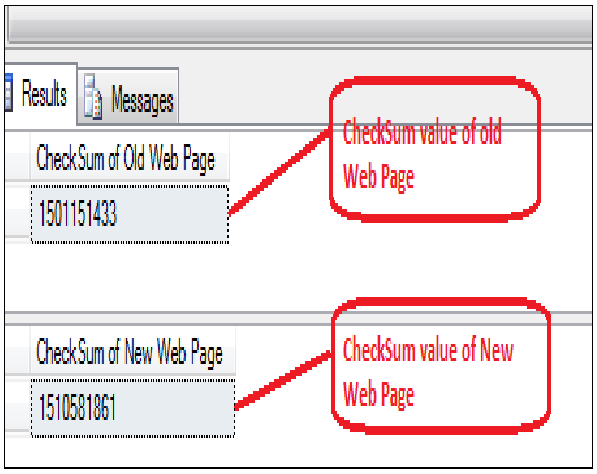
Fig.14. Initial Version of Web Page.


A novel web page change detection approach using Sql Server

The **WWW** is very dynamic in nature and is basically used for the exchange of information or data, the content of web page is continuously added, deleted, or changed, which makes Web crawlers challenging to keep up-to-date with the current version of the Web page. The web page is frequently changes its content hence it become necessary to develop an efficient system which could detect these changes efficiently in the lowest browsing time to achieve this changes. We compare the old web page hash value with the new web page hash value if hash value is changed that means web page content changed. Changes in a web page can be detected with the use of hash value which is unique.

Fig.15. Changed Version of Web Page

As we can see in Figure 16 which shows different checksum value of both above web pages, so we can say that there is changes in web pages. It also gives different value of checksum even if color of font is changed.


Results
Messages
CheckSum value of old Web Page
CheckSum value of New Web Page

CheckSum of Old Web Page
1501151433

CheckSum of New Web Page
1510581861

Fig.16. Checksum Values of Both Web Pages.

VI. Conclusion

In this paper we develop an efficient web page change detection system which is used in web page change detection. In this paper, classification of web page change is provided. This paper basically determines whether a web page has been altered from its earlier stored version or not. Change detection is refined by matching the recently retrieved web page with an already retrieved page saved on database. Change detection is taking into account computing the Hash value of two HTML documents. The web page portions which are not 100% similar are considered to have been changed. Any type of change can be detected by calculating checksum value. It gives the result very fast and takes very less time for evaluate checksum.

Acknowledgment

The author especially thanks to mention the financial help provided by MANF SRF under University Grant Commission (UGC), New Delhi.

References

[1] Cho, Junghoo, Angeles, Los, and Garcia-Molina, Hector, “Effective Page Refresh Policies for Web Crawlers”, ACM Transactions on Database Systems, Volume 28, Issue 4, pp. 390 – 426, December 2003.

[2] Lawrence, S., and Giles, C. L., “Accessibility of information on the web”, Nature, 400:107-109, 1999.

[3] Cho, J. and H. Garcia-Molina. The evolution of the Web and implications for an incremental crawler. VLDB conference 2000, 200-209, 2000.

[4] Fetterly, D., M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of Web pages. WWW ‘03, 669-678, 2003.

[5] Kim, J. K., and S. H. Lee. An empirical study of the change of Web pages. APWeb ‘05, 632-642, 2005.

[6] Koehler, W. Web page change and persistence: A four-year longitudinal study. JASIST, 53(2), 162-171, 2002.

[7] Kwon, S. H., S. H. Lee, and S. J. Kim. Effective criteria for Web page changes. In Proceedings of APWeb ’06, 837-842, 2006.

[8] Ntoulas, A., Cho, J., and Olston, C. What's new on the Web? The evolution of the Web from a search engine perspective. WWW '04, 1-12, 2004.
[9] Pitkow, J. and Pirolli, P. Life, death, and lawfulness on the electronic frontier. CHI '97, 383-390, 1997.
[10] Olston, C. and Pandey, S. Recrawl scheduling based on information longevity. WWW '08, 437-446, 2008.
[11] Selberg, E. and Etzioni, O. On the instability of Web search engines. In Proceedings of RIAO '00, 2000.
[12] Teevan, J., E. Adar, R. Jones, and M. A. Potts. Information reretrieval: repeat queries in Yahoo's logs. SIGIR '07, 151-158, 2007.
[13] Erwin Leonardi, Sourav S. Bhownick, "Detecting Content Changes on Ordered XML Documents Using Relational Databases".
[14] F.P.Cubera, D.A. Epstein. "Fast Difference and Update of XML documents", Xtech, san Jose, March 1999.
[15] H. Mayurama, K. Tamora, "Digest value for DOM (DOM hash) proposal", IBM Tokyo Research Laboratory, http://www.tri.ibm.co.jp/projects/xml/domhash.htm, 1998.
[16] G. Cobena, S. Abiteboul, A. Marian, Detecting changes in XML documents, in: 18th International Conference on Data Engineering, San Jose, CA, 2002, pp. 41–52.
[17] Y. Wang, D. DeWitt, J. Cai, X-Diff: an effective change detection algorithm for XML documents, in: International Conference on Data Engineering, Bangalore, India, 2003, pp. 519–530.
[18] J. Jacob, A. Sache, S. Chakravarthy, CX-DIFF: a change detection algorithm for XML content and change visualization for WebVigiL, Data and Knowledge Engineering, Volume 52, Issue 2, pp. 209–230, 2005.
[19] S. Flesca, E. Masciari, Efficient and effective web change detection, Data and Knowledge Engineering, Volume 46, Issue 2, pp. 203–224, 2003.
[20] H. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics, Volume 2, Issue 1, pp. 7–21, 2005.
[21] I. Khoury, R. El-Mawas, O. El-Rawas, E. Mounayar, H. Artail, An efficient web page change detection system based on an optimized Hungarian algorithm, IEEE Transactions on Knowledge and Data Engineering, Volume 19, Issue 5, pp. 599–613, 2007.
[22] Francisco-Revilla, L., Shipman, F., Furuta, R., Karadkar, U. and Arora, A., "Managing Change on the Web", In Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, pp. 67 – 76, 2001.
[23] P. L. Hegaret, R. Whitmer, and L. Wood, "W3C Document Object Model (DOM)", June 2005. URL http://www.w3.org/DOM/.
[24] HTML 4.01 Specifications. http://www.w3.org/TR/html4/.

Authors' Profiles

Md. Abu Kausar received his BCA degree from T. M. Bhagalpur University, Bhagalpur in 2002, Master in Computer Science from G. B. Pant University of Agriculture & Technology, Pantnagar, Uttrakhand, India in 2006 and MBA (IT) from Symbiosis, Pune, India in 2012. He has received Microsoft Certified Technology Specialist (MCTS).

At present, he is pursuing Ph.D in Computer Science from Jaipur National University, Jaipur, India and he is receiving UGC MANF SRF Fellowship during Ph.D Programme.

He is having 8+ years of experience as a Software Developer and research. His research interest includes Information Retrieval and Web Crawler**.**

Dr. V. S. Dhaka is a young and dynamic technocrat with 10 years of intensive experience in industry and academia. He is M.Tech and Ph.D in computer Science from Dr. B R Ambedkar University, Agra, India. With more than 32 publications in international journals and paper presentations in 27 conferences/seminars, he always strives to achieve academic excellence.

He has been awarded by the employers with "Employee of the Quarter Award", "Mentor of the year award" and with letters of appreciations for his commitment, advocacy and mentor-ship. He has organized several Conferences, Seminars and Workshops.

Dr. Sanjeev Kumar Singh is working as Assistant Professor in Department of Mathematics at Galgotias University, Gr. Noida, India. He earned his M. Sc. and Ph.D. degrees with major in Mathematics and minor in Computer Science from G.B. Pant University of Agriculture and Technology, Pantnagar, Uttrakhand, India. Before that he completed B.Sc. (Physics, Mathematics & Computer Science) from Lucknow University, Lucknow.

He is having more than nine years of teaching and research experience. Besides organizing three national conferences, he has published several research papers in various International/National journals of repute and several national and international conferences and workshops. His areas of interest include Mathematical Modeling, Differential Geometry, Computational Mathematics, data mining & Information Retrieval.