International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:4, 2012

An Optimal Algorithm for HTML Page Building Process

Maryam Jasim Abdullah, Bassim. H. Graimed, Jalal. S. Hameed

Abstract**—**Demand over web services is in growing with increases number of Web users. Web service is applied by Web application. Web application size is affected by its user's requirements and interests. Differential in requirements and interests lead to growing of Web application size. The efficient way to save store spaces for more data and information is achieved by implementing algorithms to compress the contents of Web application documents. This paper introduces an algorithm to reduce Web application size based on reduction of the contents of HTML files. It removes unimportant contents regardless of the HTML file size. The removing is not ignored any character that is predicted in the HTML building process.

Keywords**—**HTML code, HTML tag, WEB applications, Document compression, DOM tree.

List of Symbol:
HTML: Hypertext Markup Language
n: a number
DOM: Document Object Model
P: Paragraph
px : Pixels

I. Introduction

HTML document is containing code written in a high level language which mean can be understood by human [1, 8, and 9]. This priority of HTML code allowed developers to expand the content of HTML document without limitations [1]. Code or content of HTML document is an important to initial web page in response to end user request. HTML code contains a set of elements. These elements structure, content and attributes affected the HTML page view in internet browser [5]. Content expand is solved by applying compressing algorithms over content [1, 2]. Compressing algorithm is a technique used to reduce the content of HTML document to save store space and reduce size of package transfer over internet in response to end user page request [3]. Algorithm that is used to compress documents is gzip or deflate with common LZ77-based [1, 2]. This algorithm does not try to go throw the content of HTML document and analyses its structure. Algorithm works to replace duplication strings with pointer refer to string location which similar to in the HTML document content. Our proposed algorithm goes throw the content of HTML document, builds DOM tree, removes characters and spaces between elements, and saves each node in DOM tress as object related to HTML page. DOM tree presented the structure of HTML page [6, 7, and 4].

Maryam Jasim Abdullah is with the website unit, University of Baghdad, Al-Jadria, Baghdad, Iraq (e-mail: m.jabdullah@yahoo.com).
Bassim. H. Graimed is with the website unit, University of Baghdad, Al-Jadria, Baghdad, Iraq (e-mail: bassim@uobaghdad.edu.iq).
Jalal. S. Hameed is with the website unit, University of Baghdad, Al-Jadria, Baghdad, Iraq (e-mail: jalal.hameed@uobaghdad.edu.iq).

In response to end user request the algorithm use objects related to requested HTML page to build it. Simulation results show that the proposed algorithm for HTML documents is reduced the HTML file size to 7.22%.

The structure of paper is as follows: Section 2 reviews HTML code overview. In section 3 an algorithm commonly used to compress documents is discussed. Detailed steps of compression over documents in the proposed algorithm are presented in section 4. In section 5 simulation results of the proposed algorithm and a comparison with the common algorithms are made. Section 6 provides conclusion and summery of the work.

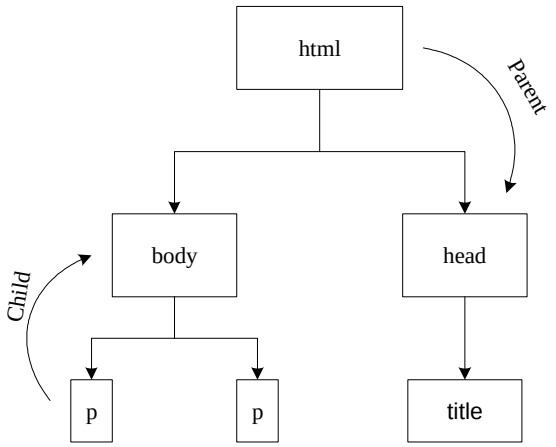

Fig. 1 DOM tree to structure HTML Elements.

II. HTML Code

Hypertext Markup Language is language use to write a code for a web page [1, 8, and 9]. HTML instructions exist within tag. Each web page consists of n-tags. Tags represent the structure of the web page and its view to the end user.
There are two types of tags:

A. Tag with a content.

Tag with content consists of three parts: start tag, content, and end tag.

Start tag starts with the lesser than (<) character, follows by element and its attributes, and it is ended with greater than (>) character (e.g. <body style="font-size: 18px ;">).

End tag starts with the lesser than (<) character, follows by forward slash (/) character and element, and it is ended with greater than (>) character (e.g. </body>).

The content of tag is between the start tag and end tag (e.g. First paragraph).

B. Tag without a content.

Tag without content consists of one part which starts with the lesser than (<) character, follows by element and forward slash (/) character, and it is ended with greater than (>) character (e.g.
).

III. LZ77-Based Algorithm

Gzip utility is used with LZ77-based algorithm to compress files [1, 2]. LZ77-based algorithm is used Huffman coding to generate a series of compressed blocks and to find the location of Strings duplication. It replaces strings with a pointer refer to the location of a similar string. The size of strings duplication starts with 32k bytes to 258 bytes.

Block content is a tree code which represents the data file structure and hold compressed data [2]. There are two types of data which are contained by compressed data:

- Literal bytes: for data string that are not found for its similarity strings.
- Pointers to duplicated strings: contain two value length of the duplication string, and backward location of parent.

IV. Setting Tag Objects For Html Page

This section presents how our algorithm arranges each tag in HTML page as object to reduce page size. The following example for HTML code is containing six elements with attributes and content. This HTML code is used as input to explain each steps in our algorithm execution.

HTML code

```
<html>
        <head>
                < title > Sample of code </title >
        </head>
        <body style="font-size: 18px ;">
                <p style="color: #00b050 ;"> First paragraph</p>
                <p> Second paragraph </p>
        </body>
</html>
```

A. Building a DOM Tree

DOM tree consists of many nodes. Each node is parent from zero-to-n child [6, 7, and 4]. Parent node connects to its child by adage. DOM tree for HTML page represents its structure and each node in DOM tree holds one tag of HTML page.

HTML code example contains six tags. As it can see in Figure.1, The DOM tree for the HTML code contains six nodes which are: html, head, title, body, and two p nodes. The <HTML> tag is a parent to the entire tags in the HTML page code, <head> tag is parent for < title > tag, and <body> tag is parent for two <p> tags.

B. Creating a Tag Object

Tag object is a node in DOM tree which saves all information need to present tag in the view of HTML page. The following are Tag object variables:

- Element.
- Parent: Tag object.
- Sequence: Tag objects order of view between its parent children.
- Attribute: Saving values of tag object attributes.
- Page: Name of HTML page which is tag related to it.
- Content: Data containing by tag to present in HTML page.

The pseudo-code below is written to find tag and their variables. Element and tag attributes are located in the start tag (example <body style="font-size: 18px ;">). Content of tag is located between start and end of tag. (Example < title > Sample of Code </ title >, the content of < title > tag is “Sample of Code” which is between < title > and </ title >). The pseudo-code extracts element, tag attributes, and tag content variables of object tag.

```
if(HTMLCodeText.length()>0)
 {
start = HTMLCodeText.indexOf("<");
        end= HTMLCodeText.indexOf(">");
element=HTMLCodeText.substring
(start+1,HTMLCodeText.indexOf(Next Space).trim();
SetTagAttributes(HTMLCodeText.indexOf(Space),end);
HTMLCodeText=HTMLCodeText.substring (end+1).trim();
end =this.getText().indexOf("<");
        ContentOfTag=HTMLCodeText.substring (0,
        end).trim();
        HTMLCodeText=HTMLCodeText.substring.substrin
        g(end).trim();
              }
```

Parent and sequence values are affected by opening and closing tags. The steps to extract parent and sequence values for each tag object in HTPL page are illustrated in Fig.2. Steps are as following:

- Flowchart is starting by checking the HTML code text length. If HTML code text length equals “0”, the function is ended. Otherwise, go to set end variable to equal index of first ">" character find in HTML code text and forward slash to equal index Of first "/" character find in HTML code text.
- If forward slash is less than end, that mean tag is now closed. In other mean tag now is not further parent for any next open tag. Then removed end tag from HTML code text, set parent to be equal to parent of closed tag which is still open and its end tag have not get yet, and set sequence to be at value of high tag sequence in parent tags children and increasing it by 1.
- Continue in comparison until end is less than forward slash. That means new tag is open.
- Create a new tag object and set its variables element, tag attributes, tag content, parent, and sequence.
- Sequence is increasing. By created new object in pervious step a new child was added to the parent. So next child order must increase by 1.

- Checking HTML code text length is continuing until adding all tag objects with their variables.

C.Tag Object for Code Size Reduction

Our algorithm depends on saving each tag with their variables in one object. All Characters and spaces which can be added back to its location during the building process of the page in request time are removed. The objects for HTML pages are grouped in one document to minimize the size of HTML pages for one web application.

By applying our algorithm over HTML code six tag objects are created. Figure.3. shows that each object holds values. These values represent tag view in the HTML page in request time. Total number of characters is reduced from 143 to 79 characters in the HTML code text. This reduction affects the total size of the web application if we apply it over all HTML pages and save all object created for each page in one document.

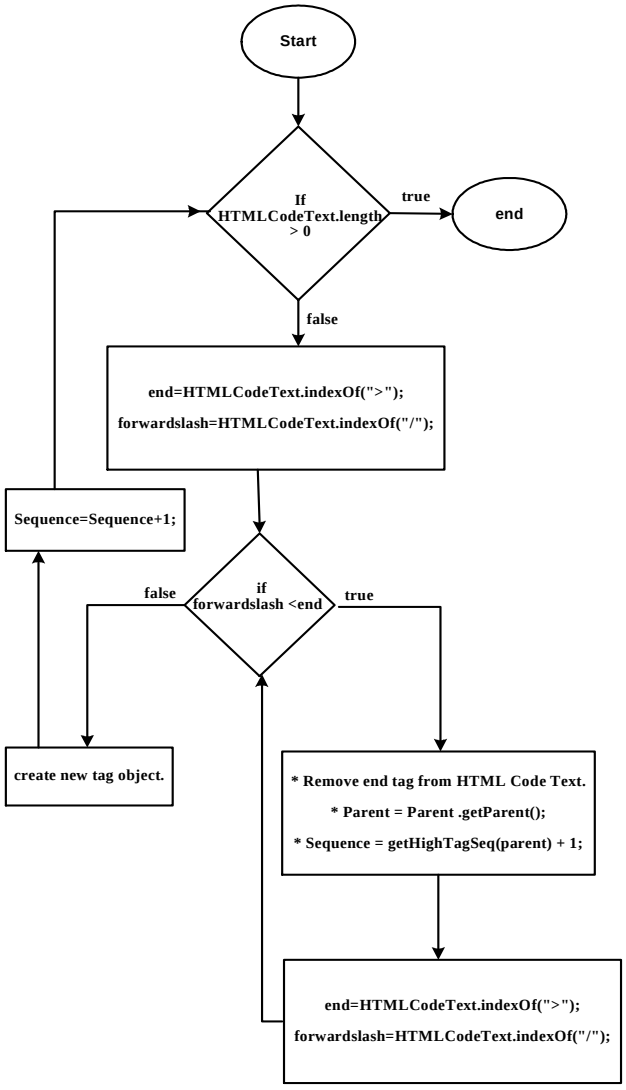

Fig. 2 Flowchart to find parent and sequence value for each tag object in the HTML code

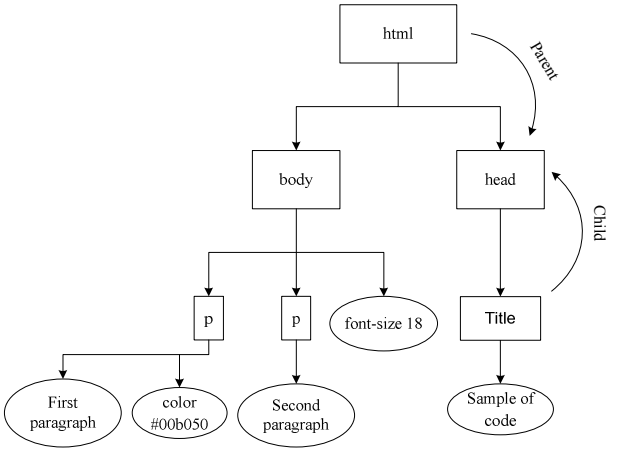

Fig. 3 HTML code containing six tag objects.

D.Page Request

By applying our algorithm over HTML code documents for web application and create all objects for each HTML documents, objects created for all HTML are saved in one document source in the server side. The following steps are to create HTML page request by end user:

- When end user request for page, page name is save as request parameter.
- Our algorithm uses page name parameter in the request to get all tag objects related to the page and save them in list.
- Sort tag list. First tag in the sorted list will be <html> tag following by its children. Children are sorted depends on value of sequence variable for each child. Each child is parent for next level of tags. So each tag object is parent follow by its children and these children are sorted depends on values of sequence variable for each child.

The following pseudo-code is a function uses the sorted tag list to generate HTML page:

```
Function String build-HTML-Page (List Tag-obj-list)
        String Page="";
        For all tag in Tag-obj-list
                Page=page+ '<' +tag.getElement ();
                Page=page +getTag Attributes(tag);
                Page=page + '>' ;
                Page=page + '<' +tag.getContent();
                Page=page + '</' +tag.getType()+">";
          Return page;
```

String page generated in this function is a complete HTML page.

V.Algorithms Comparison

To show the high performance of our proposed algorithm, a comparison is made with common LZ77 algorithm. In the LZ77 algorithm, the compression of HTML files depends on finding string duplication and to replace them with a pointer refer to its parent string that is similar to it. While our algorithm technique goes throw the HTML cod structure and

remove all characters and spaces which their locations can be predicted in process of building HTML page on time of end user request.

LZ77-based algorithm starts to search for duplication string with size of 32 K bytes. The size is reduced to 258 bytes if the similarity is not found. LZ77-based algorithm will not have any compression effect over HTML file if its size is less than (258 *2) bytes [2]. A sample HTML file with 83 bytes size is taken for the comparison.

The proposed algorithm in this work algorithm generates file that is gathering all important information for each tag in the HTML file with size of 77 bytes. This is showing that the size of HTML file is reduced by 7.22% of the total size of HTML file. LZ77-based algorithm will not have any effect over that HTML file of 83 bytes size because the lowest HTML file size for duplication string is 258 bytes. The proposed algorithm in this paper is affected on the size of HTML file by removing characters and spaces regardless of the size of HTML file. This removing is not ignored any character that is predicted in the HTML building process.

The proposed algorithm is created one document source for all HTML pages. One source for read and write HTML pages. In request time, there is a read of all tags for HTML page from source document to build HTML page. In response, all updates over tags for HTML page are saved on the same document.

Our algorithm is use to represent each tags of HTML page as object. This gives HTML page the ability to restructure its tags easily. That means, we can easily add, remove, and update tag object.

VI. Conclusions

In this paper, an optimum algorithm is proposed to compress HTML documents for web application. The proposed algorithm generates file that is gathering all important information for each tag in the HTML file. This is leading to an improvement in the HTML building process by reduction the total size of HTML file. The high performance of the proposed algorithm is provided by removing characters and spaces with no ignoring of any character that is predicted in the HTML building process.

References

[1] Przemyslaw Skibinsk : Improving HTML Compression. Data Compression Conference, 2008. DCC 2008 , Page(s): 545.
[2] Deutsch, P.: DEFLATE Compressed Data Format Specification version 1.3. RFC1951, (1996), http://www.ietf.org/rfc/rfc1951.txt.
[3] Nielsen H.F.: HTTP Performance Overview,2003, http://www.w3.org/Protocols/HTTP/Performance/.
[4] Document Object Model (DOM) Level 2 Core Specification, Version 1.0, W3C Recommendation 13 November, 2000. http://www.w3.org/TR/DOM-Level-2-Core/Overview.html.
[5] Matthijs, N. "HTML, the Foundation of the Web," March, 2008. [Online]. Available: http://www.wpdfd.com/issues/86/html_the_foundation_of_the_web/. [Accessed: Aug. 10, 2010].
[6] Chakrabarti. S. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In WWW10, Hong Kong, May 2001. Online at http://www10.org/cdrom/papers/489.
[7] Rozinajová.V and Hluchý.O, "One approach to HTML wrappers creation: using Document Object Model tree", in Proc. CompSysTech, 2009, pp.41-41.
[8] HTML code tutorial, "Document Tags ". URL http://www.htmlcodetutorial.com/document/.
[9] Web Source, "HTML Tags / Codes / Web Page Design". URL http://www.web-source.net/html_codes_chart.htm.