Available online at www.sciencedirect.com

ScienceDirect

Energy Procedia 00 (2012) 000–000

Energy Procedia

www.elsevier.com/locate/procedia

A WEB SITE ENERGY SAVING DEPLOYMENT FOR WEB-BASED HTML USING DESIGN COHESION

Wipharat Pitpumwiti[1], Peraphon Sophatsathit[2], Achara Chandrachai[1]

[1]Technopreneurship and innovation Management program, [2]Dept. of Mathematics and Computer science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand
Wipharat.P@Student.chula.ac.th, Peraphon.S@chula.ac.th, achandrachai@gmail.com

Abstract

The objective of this research is to minimize the electricity consumed by individual web site using cohesion technique. The underlying principle lies in rearrangement of web-based HTML code in compliance with W3C and cohesion principles. As such, this research exemplifies good programming discipline in system design which yields considerable energy saving. The result suggests a simple approach for energy consumption environmental green computing system that depends on how efficient one manages the supporting processing environment.

Keywords; Cohesion, HTML, energy consumption, green computing, web-site.

1. Introduction

The explosion of 18.3 million internet user stimulates enormous traffic which calls for more web site development and support. As more user accesses mean more computations, the need for higher power consumption grows. The number of internet access in Thailand alone based on truehit.net is 125 million page views [4]. Statistics in 2011 from one popular website, sanook.com, average 15 million page views [1] per day, the period taking 1.817 seconds [2]. Small as the number reveals, the cumulative figure across country yields an unprecedented cost of electricity consumption that will be portrayed in subsequent sections. Moreover, 80% of the information broadcasted on the web site is designed based on back-end supporting the database, where data sources consist of HTML code linking to various artifacts and supplementary pages [8].

The objective of this research is to propose an energy conserving HTML page processing by categorizing commands into sub-module to reorder HTML code structure based on cohesion principles. The rationale behind such rearrangements is evident from researches on the environment of code processing. The better code organization in compliance with complier mandate, the more efficient/lower rate of power consumption to access memory [9].

This paper is organized as follows. Section 2 briefly describes fundamentals of cohesion. Section 3 narrates how cohesion is applied to HTML. Section 4 presents research methodology whose results are summarized in Section 5. The paper concludes with some final thoughts in Section 6.

2. Cohesion

Bieman and Ott [7] define cohesion as a unit of software or module referring to the relationship of components inside an enclosing software module. A software module with strong connection is the module of basic function that cannot be separated. Thus, functional cohesion [3] will be used as the principal design guidelines.

3. Structure of Cohesive HTML Source

One of the essential mandates to rearrange the HTML source is to adhere to correctly parsed code. Due to versatility of the language, we resorted to retain all artifacts external or beyond the scope of this work as is, for example, php blocks, frame construct, etc. In order to preserve the correctness and intention of the original code, the proposed approach confines its code transformation to the following stipulations:

1. Style and its accompanying properties are converted external stylesheet,
2. All inputs are set up by site administrator,
3. Block and inline elements are separated, and
4. All unsupported features are left unchanged.

The following HTML constructs [6] are procedurally rearranged by design cohesion.

3.1. Block element

Block-level elements [10] consist of many components built for self-identification. If they are found as ordinary sentences, they will be grouped in paragraph. Text-level elements represent the message inside the encompassing component of that block. The following three types of block elements are supported:

- Structural block: <ol> <ul> <dl> <table> <tr> <tbody> <col>
- multi-purpose block: <div> <li> <dd> <td> <th> <form>
- terminal block: <h1> <p> <dt> <caption> <address> <blockquote>

3.2. Inline element

Inline elements are processed to identify the meaning and flow of text, and to insert external content into the document [10]. They possess some specific distinctions from block-level elements [11]:

- Inline elements generally only contain text, data, or other inline elements.
- Inline elements do not generally begin new lines of text.
- Inline-semantic encompassing text intermingled with zero or more of the following elements:
  - Importance: <span> <em> <strong>
- Inline-block including replaced elements and form controls:
  - Replaced: <img> <object> <embed>
  - Controls: <input> <textarea> <select> <button> <label>

3.3. External Stylesheet

External style sheet is written as an external HTML file in the form of CSS for display processing. The system will check the opening tag to generate CSS file using the same name as the original HTML file. Attributes are created by preceding each class name with 'style_' followed by a number, input attributes and corresponding value pointer to the database. This simple naming avoids duplication of class name clash. An example inline CSS will result in a new class beginning with <div style=”background: url(‘img/image.jpg’);”> </div> becomes <div class=”style_1”></div>

The above constructs are examined to identify the functionality of each code segment and regroup as an HTML module in accordance with the cohesion category. The tighter is the cohesive property, the better the code is organized.

4. Research Methodology

The proposed algorithm converts original HTML code to the designated cohesive code by rearranging mixed HTML statements into smaller HTML and CSS files. The process is shown in Figure 1.

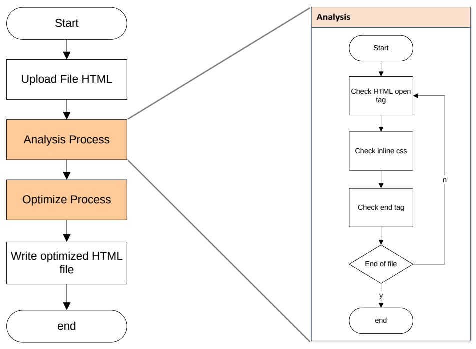

Fig. 1. Overview of the proposed algorithm

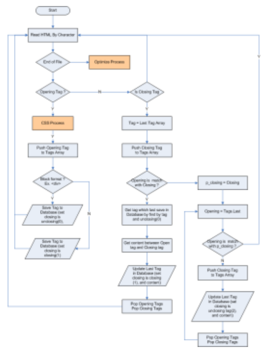
Start
Read HTML By Character
End of File
Optimize Process
Opening Tag ?
N
Is Closing Tag
Y
CSS Process
Tag = Last Tag Array
Push Opening Tag to Tags Array
Push Closing Tag to Tags Array
Block format ? Ex. <div>
Opening is match with Closing ?
p_closing = Closing
Save Tag to Database (set closing is unclosing(0))
Opening = Tags Last
Save Tag to Database (set closing is closing(1))
Get content between Open tag and Closing tag
Opening is match with p_closing ?
Update Last Tag in Database (set closing is closing (1), and content)
Push Closing Tag to Tags Array
Update Last Tag in Database (set closing is unclosing tag(2), and content)
Pop Opening Tags Pop Closing Tags

Fig. 2. Procedural analysis of web-based HTML transformation

Figure 1 illustrates the proposed algorithms with the emphasis on analysis of cohesion to support the relationship and independence among modules. Figure 2 elaborates the decomposition when an HTML open tag is located, the system will store the data and pointer to identify the open tag in the database for rechecking the condition assigned. Then the system will find inline open tag CSS to generate the corresponding CSS file under the same name as the HTML file, store the accompanying attributes in this CSS file, and archive each tag position in the database for subsequent retrieval. This process repeats until the matching close tags are located. Any unmatched open tags will be marked as no close tag. After checking the original code, the system will retrieve the following stored data for further processing.

- HTML tag with no matching close tag
- HTML tag with inline CSS
- HTML tag with correct condition

The above data are reassembled into new HTML files having the same name based on predetermined cohesion stipulation. In so doing, the original files are not affected and can be used for subsequent correctness verification. The resulting new files may become smaller or larger than the original files, depending on how the components are processed.

One additional provision for future extension is the use of case HTML based on tight cohesion to support any features or capabilities needed in post-production and other application software integration. Our tool furnishes some precautions to prevent security breach by restricting such extension support to be handled by site administrator.

5. Experimental Results

The fact that HTML code is relatively small compared with conventional programs complicates the test process somewhat. For every newly web page access, execution takes place at the first invocation only which consumes little power. Subsequent browsing virtually consumes no power as the content has already resided in cache. Thus, we had to conduct a sizable number of page views in order to obtain any "measurable" discrepancies of power consumption. Fortunately, all selected web sites are heavily visited on a daily basis. On the national scale, the savings of power consumption is highly remarkable. We conducted the experiment in two different environmental settings, namely, single isolated user and multi-user environments to demonstrate the disparity of usage. We have developed a software tool based on WinMerge to carry out the above procedural transformation. Note that the size of original file and optimized one are 72 KB and 74 KB, respectively. However, the size increase is offset by the electrical cost decrease.

The experiments were run on Intel(R) Celeron(R) CPU E330@2.5GHz. The sample programs can be accessed from http://www.wp-greenpage.com. Table 1 and 2 show monthly comparative statistics of web processing under single isolated user environment and multi-user environment, respectively.

Table 1 Single isolated user environment

website	Page views [4]	(before) Process time per seconds	(after) Process time per seconds	Unit per month (kw)	(before) electricity consumption per month (Baht)	(after) electricity consumption per month (Baht)	Difference per seconds	Original size (KB)	Result size (KB)
Sanook	768,603.70	1.08	0.80	449.63	1,769.80	1,244.38	- 0.28	162	162
Kapook	647,943.00	1.18	1.14	414.14	1,630.11	1,574.85	- 0.04	290	247
Mthai	663,383.03	1.04	1.03	373.70	3,750.00	3,713.95	- 0.01	307	280
Dek-d	402,590.70	1.39	1.32	303.11	1,132.50	1,075.47	- 0.07	459	430
manager	351,754.23	0.71	0.68	135.27	437.15	418.68	- 0.03	103	143
teenee	262,126.67	0.94	0.90	133.46	431.29	412.94	- 0.04	262	238
weloveshopping	233,752.83	1.07	1.06	135.47	437.80	433.70	- 0.01	319	300
oknation	200,780.90	0.79	0.76	85.91	269.61	259.37	- 0.03	142	178
siamzone	195,359.47	0.73	0.70	77.24	242.41	232.45	- 0.03	99	90
siamha	192,724.40	0.55	0.51	57.41	180.17	167.07	- 0.04	140	110

Table 2 Single isolated user environment

website	Page views [4]	(before) Process time per seconds	(after) Process time per seconds	Unit per month (kw)	(before) electricity consumption per month (Baht)	(after) electricity consumption per month (Baht)	Difference per seconds	Original size (KB)	Result size (KB)
Sanook	768,603.70	1.00	0.80	416.32	1,638.70	1,310.96	-327.74	162	162
Kapook	647,943.00	1.14	1.10	400.10	1,574.85	1,442.41	-132.44	290	247
Mthai	663,383.03	1.05	1.03	377.29	1,409.66	1,382.81	-26.85	307	280
Dek-d	402,590.70	1.29	1.25	281.31	1,051.03	1,018.44	-32.59	459	430
manager	351,754.23	0.71	0.69	135.27	437.15	424.83	-12.32	103	143
teenee	262,126.67	0.97	0.93	137.72	445.06	426.70	-18.36	262	238
weloveshopping	233,752.83	1.03	1.00	130.41	421.43	409.15	-12.28	319	300
oknation	200,780.90	0.78	0.72	84.82	266.20	245.72	-20.48	142	178
siamzone	195,359.47	0.69	0.54	73.01	229.13	179.31	-49.82	99	90
siamha	192,724.40	0.48	0.40	50.10	157.24	131.03	-26.21	140	110

The formula to calculate electricity consumption [5] from the processing site are shown below,

*KW per day = watt * (hour * page views)/1000*

*KW per year = watt * (hour * (page views * 365))/1000*

*Electricity consumption = KW * Service rates*

It is apparent that energy savings are prevalent across the board. Savings of multi-user environment are higher than the single-isolated user environment counterpart because of contribution from page sharing from cache, which is shown in Figure 3 and 4, respectively. However, the size of the transformed source may become larger than the original source as depicted in Figure 5.

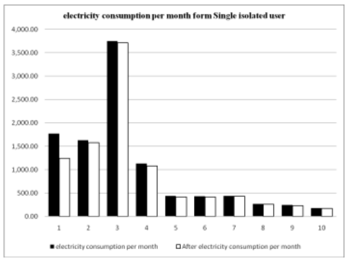

Fig. 3. Monthly electricity consumption from single isolated user environment.

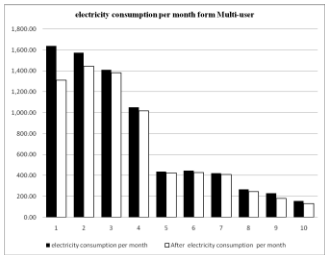

Fig. 4. Monthly electricity consumption from multi-user environment.

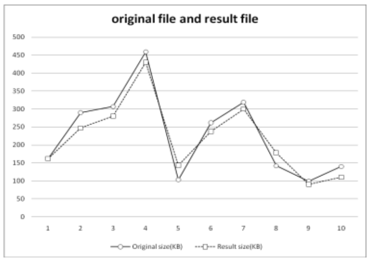

Fig. 5. Size of original and result files.

We conducted a preliminary user's survey and found 72% would prefer visiting web sites that have an energy saving indicator, while 56% would like to see electricity usage meter. On the other hand, 56.4% of web site developers regard web page processing to be the most important factor of energy saving scheme. A proportional 23.6% consider strict HTML compliance with W3C will help conserve computation power.

6. Conclusion

This paper presents a concise and straightforward approach to reduce power consumption required by web-based HTML processing. The underlying principle is rearrangement of HTML code accord to design cohesion and language standard. Despite the infinitesimal electricity consumption savings from the modified/optimized HTML code, the enormity scale of use amplifies such savings to be worthy of investigation, as evident from our findings. One might contend that the fast pace of technological advancement will render HTML to be a thing of the past. The research fundamental philosophy still remains valid—good program design goes a long way. Not only innumerable technical merits can precipitate from such an effort, but also the ultimate green technology will benefit all mankind and the planet as a whole.

References

[1] Piya Tanthawichian (2006). Thailand Internet Snapshot 2010 by Truehits.net Statistics (Online). http://truehits.net/awards2010/download/truehits-awards2010-20May2011.pdf, accessed on July 08, 2011.

[2] Alexa Internet (2011). Inc. Statistics Summary for sanook.com (Online). http://www.alexa.com/siteinfo/sanook.com, accessed on Nov 24, 2011.

[3] J.M. Bieman and Kang Byung-Kyoo, Measuring design-level cohesion, *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 111 - 124, Feb. 1998.

[4] Truehits (2006). Index Regular Category (Online). http://directory.truehits.net/graph/linegraph5.php, accessed on Nov 29, 2011.

[5] PEA-Tariff (2004). Provincial Electricity Authority (Online). http://www.pea.co.th/rates/Rate2011.pdf, accessed on Nov 29, 2011.

[6] Web Design Group (1996-2006). Overview of all HTML elements (Online). http://htmlhelp.com, accessed on Nov 29, 2011.

[7] J.M. Bieman and L.M. Ott, Measuring functional cohesion, *IEEE Transactions on Software Engineering*, vol. 20, no. 8, pp. 644-657, Aug. 1994.

[8] Arnaud Sahuguet and Fabien Azavant, "Web Ecology:Recycling HTML pages as XML documents using W4F", University of Pennsylvania Scholarly Commons, Jan. 1999.

[9] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto, The impact of source code transformations on software power and energy consumption, J. Circuits Systems & Computers, vol. 11, no. 5, pp. 477-502, May 2002.

[10] Michael Bowers, Dionysios and Victor Sumner, *Pro HTML5 and CSS3 Design Patterns*. 1st ed. Apress. 2011.

[11] Jennifer Kyrnin (2011). inline elements (Online). http://webdesign.about.com/od/htmltags/g/bldefinlineelem.html, accessed on Dec 23, 2011.