# Adaptive Process Management System through Biological Web Log Mining

Heung Ki Lee [1], Jaehee Jung [1], Gangman Yi [2]

[1.] Samsung Electronics, Suwon, Korea
heungkilee@gmail.com, cleme76@gmail.com
[2.] Department of Computer Science & Engineering, Gangneung-Wonju National University, Korea
gangman@cs.gwnu.ac.kr

**Abstract:** Main memory management is critical for enhancing the performance of web server systems that include biological information. For decreasing the transaction time of incoming requests from users, such systems create several processes for future requests, saving the time needed to create processes that handle incoming requests from users. However, inefficient process management can decrease the performance of web server systems, and web cache systems connecting through proxy servers create dynamic access patterns that make it difficult to predict how many requests are coming into a system. Furthermore, while persistent and pipeline schemes decrease transaction time of incoming requests by sending multiple requests at the same time, these schemes waste available memory space by requiring multiple processes in order to handle multiple connections. Too many active processes result in a reduction of the system's overall performance. Therefore, we suggest an adaptive process management scheme through web log mining. In our scheme, the numbers of web processes are controlled through prediction of incoming requests. Our management of processes saves on available memory without decreasing transaction time. We also demonstrate the effectiveness of our scheme through application to real web workload.

[Lee H.K, Jung J, Yi G. **Adaptive Process Management System through Biological Web Log Mining**. *Life Sci J* 2014;11(7):716-720] (ISSN:1097-8135). http://www.lifesciencesite.com. 104

**Keywords:** Web Log Mining; Web Server Systems, Biological web pipeline

## 1. Introduction

The introduction of smartphones has made it possible for a user to access the Internet at any time. However, as a result this has dramatically increased network traffic. Even though web server systems have gained increased performance, users are still not satisfied with web service. A single web document usually requires several pieces of information, including html, text, images, and video, and in general, each piece of information is stored separately. To increase response time, all of the information associated with the requested document should be provided to users as soon as possible. Therefore, it is critical to increase the throughput of web server systems. If one piece of information is delayed, the entire requested web document is delayed.

Modern web browsers increase performance by establishing multiple connections with a server. When a user requests a document, the browser retrieves the main object from the server. After receiving the requested main object, including html and jsp, the browser analyzes it and extracts additional information. To obtain this additional information as soon as possible, the browser opens more connections with the server. In figure 1, a user requests 'a.htm' from a server. After receiving 'a.htm', the browser extracts a list of embedded objects that are required for representing the document to the user. The requested 'b.jpg' and 'c.mov' are not available in the browser's cache, but 'd.eps' is available. As a result, the browser reads 'd.eps' from its own cache, and establishes another HTTP connection with the server to obtain 'b.jpg' and 'c.mov' simultaneously.

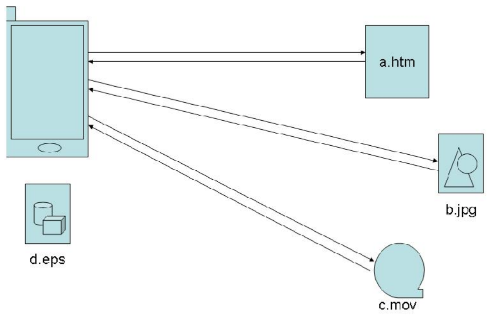

Figure 1. Retrieval of Web Document

To increase performance, the web server system creates processes before receiving requests from the user. With too many processes, the resources of the server are wasted, reducing performance. However, with too few processes, the retrieval of objects is delayed, slowing response time. To optimize performance, the server should maintain the proper number of processes for incoming requests. To allow this, we suggest ConWebPro, a system that predicts the number of requests from each web browser, and thus increases performance according to the structure of each document.

## 2. Related Works

For enhancing the performance of web servers, there are also other management schemes for web requests. Two popular schemes are web pipeline and web cache.

### 2.1 Background

#### 2.1.1 Persistent Connection and Web Pipeline Scheme

A persistent connection and web pipeline scheme decreases the overall transaction time necessary for a user to obtain information from a web server. First, under the persistent connection scheme, the browser creates a persistent connection with the web server, allowing the browser to send several requests to the server over a single connection. If the server provides only one object per connection, the browser must open and close the connection with the server for each embedded object. However, this scheme can still save the time needed to initially create the connection between server and client.

Second, the web pipeline scheme can save transaction time by creating multiple connections. If the browser only creates a single connection, it is forced to read objects sequentially, receiving each from the server one by one. However, if the browser instead creates multiple connections with a server, it can then request several embedded objects in parallel. When several requests to the server are required for one document, the web pipeline scheme enhances the throughput of the server.

#### 2.1.2 Dynamic Access Pattern

In the structure of a web object, a browser accesses the list of embedded objects after accessing the main object. However, the incoming access pattern for web objects is still dynamic. When a browser requests embedded objects, some are provided by a proxy server, not the web server. Proxy servers hold cached web objects, and when these objects are requested again, the proxy server provides them to the browsers. Requests for the cached web object are therefore not transmitted to the web server, releasing overhead.

However, the access pattern through a proxy server is dynamic, because requests to cached web objects are not accessible to the web server, making it difficult for the web server to decide how many requests are coming from browsers. In Table 1, we see which embedded objects are requested after requesting the main object from a page at the department of Computer Science and Engineering at Texas A&M University. Five embedded objects are requested when the main object is requested, but other web objects are not usually requested during a request for the main object.

Table 1: Frequency of Request for Embedded Objects

| Index | Name | Request |
|---|---|---|
| 1 | /html4/front.css | 517 |
| 2 | /html4/global.css | 517 |
| 3 | /images/bin.jpg | 477 |
| 4 | /images/header.jpg | 473 |
| 5 | /images/LOOK.gif | 446 |
| 6 | /images/random/01 | 75 |
| 7 | /images/random/06 | 69 |
| 8 | /images/random/07 | 66 |
| 9 | /images/random/09 | 66 |

### 2.2 Previous Works

There have been many previous attempts to predict the next requests from users. We classify examples here into two categories: chaining schemes based on Markov models and grouping schemes based on clusters of web objects.

Chaining schemes are based on nth Markov models. A higher order of Markov model can provide more accurate predictions, but increasing the order also increases the complexity of the prediction scheme. As a result, chaining schemes restrict the order of the Markov model. Some schemes [1], [2], [3] use the top-n related objects to predict the next requests. Other schemes [4], [5], [6]. [2] use long access sequences. [7] designed dynamic PPM models.

Grouping schemes form clusters of web objects, and then predict a group of web objects for the next incoming requests. [8] and [9] provide a caching policy for a Content Distribution Network platform. [10] and [11] design a caching policy for mobile environments. [12] provides a prediction scheme using folder structure. [13] suggests a divide-and-merge scheme via a hybrid of top-down and bottom-up schemes. [14] designs a proxy model for prefetching embedded objects. [15] uses a vector model and semantic power for a web cluster system.

Hybrid schemes design a prediction scheme based on both Markov models and grouping schemes. [16] suggests prediction schemes based on several models at the same time, including Markov models, association rules and grouping schemes. [17] uses an abstraction scheme for defining access patterns, and defines user access paths through a Markov model. [18] generates a group of access patterns to web objects using a K-means cluster scheme.

Although many examples of research provide prediction schemes for incoming requests, they do not provide process management schemes for modern web frameworks.

## 3. Web Process Management through Log Mining

### 3.1 Prediction Scheme

[19] designs the web transaction prediction scheme called a Double PPM Scheme (DPS). When browsers request main objects, web servers also receive requests to embedded objects. Therefore, we create a prediction scheme based on a grouping of one main web object and related embedded objects.

In figure 2, DPS predicts relationship between web objects at several steps. At the first step, DPS distinguishes between main and embedded objects, classifying objects based on their name. For example, when the name of an object includes ’html’, ’php’ or ’jsp’, it is likely a main object, as usually such requests are web documents. However, when the name of an object contains ’jpg’, ’mpg’ or ’ogg’, it is likely an embedded object.

In the second step, DPS creates relationships between objects. Gray circles in figure 2 indicate main objects, while white circles indicate related embedded objects. Arrows between circles show how frequently two web objects are accessed together in a single session. This demonstrates relationships within a single web document.

Continuing in figure 2, we see that this instance of DPS creates three groups: ’A’, ’B’ and ’C’. There are two different access patterns in this log file. After accessing the document ’A’, some users access document ’B’ followed by document ’C’, while other users access document ’C’ followed by document ’B’.

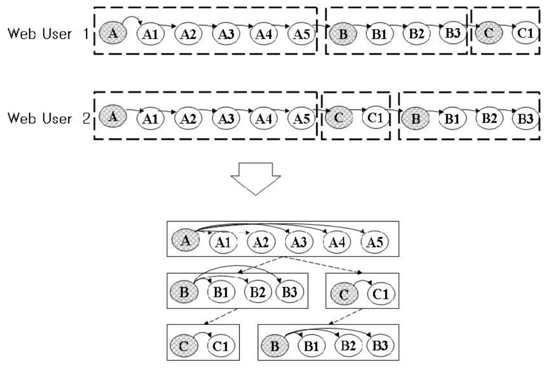
Web User 1
A A1 A2 A3 A4 A5 B B1 B2 B3 C C1
Web User 2
A A1 A2 A3 A4 A5 C C1 B B1 B2 B3
A A1 A2 A3 A4 A5
B B1 B2 B3
C C1
C C1
B B1 B2 B3

Figure 2: Double PPM Scheme

### 3.2 Web Process Management

ConWebPro decides the number of web processes to run at any time based on the access patterns of web objects. DPS finds different access patterns based on the structure of web objects. If one document contains many embedded objects, the browser will be sure to access the server after accessing the main object. Similarly, when a requested web document includes dynamic or frequently changed web objects, browsers create more HTTP connections to obtain these related embedded objects.

In analyzing the results of web log mining, ConWebPro classified web documents according to their number of embedded objects. In figure 3, ConWebPro creates three groups depending on the number of related embedded objects. Where the web server usually creates N processes for handling incoming requests, the first group includes web documents which contains more than N embedded web objects, the second group contains web documents which contain between N and N/2 web objects, and the last group contains web documents with fewer than N/2 web objects.

If the requested main object is in the first group, the web server doubles the number of running web processes for handling incoming requests. If it is in the second group, the web server creates N additional processes. If it is in the third group, the web server maintains N total processes.

## 4. Performance Evaluation

### 4.1 System Configuration in Simulation

We demonstrate performance of ConWebPro by applying it to real web traces collected over the course of two days. Based on DPS in [19], we obtain relationships between web objects using the first day’s web traces. Then, ConWebPro classified web documents into three groups depending on the number of embedded objects.

Table 2 shows real traces from web sites including the Department of Computer Science and Engineering in Texas A&M University from [19]. ConWebPro obtains relationship between web objects based on 'Day 1'. Based on the result of relationship, ConWebPro creates web processes at 'Day 2'. Web browser retrieves web objects through persistent connection and pipeline on HTTP 1.1. Therefore, web browser retrieves multiple embedded web objects simultaneously.

.

Table 2: Requests to Embedded Objects

| Name | Day 1 | Day 2 | HTTP |
|---|---|---|---|
| CS TAMU | 25479 | 20018 | HTTP 1.1 |

### 4.2 Evaluation Results

To evaluate the performance of ConWebPro, we comapare two schemes on an Apache web server. Usually, an Apache web server a static number of web processes to handle incoming requests from web users. This inefficient process management scheme wastes the available memory of the web server, causing the performance of the web server to drop.

**Step 1: Analysis on Web Objects**

ConWebPro obtains structure of web objects based on 'Day 1' web log file. At first step, ConWebPro makes groups of web requests from same web user. Web log file contains whole web requests from every web user. If multiple web users access to web server at same time, it is difficult to detect the relationship between web objects. ConWebPro extracting requests from same user based on client IP address at web log.

At second step, ConWebPro classifies web objects into main web objects and embedded web objects through path and access time. In general, web user requests 'html' or 'script' documents to web server. ConWebPro classi_es 'html' and 'script' documents into main web object. Also, embedded web objects are requested at same time after replying main web objects. ConWebPro classifies web objects into embedded web objects that are requested at same time after requesting main web object.

Lastly, ConWebPro obtains relationship based on frequency of web objects. At requesting main web object, ConWebPro checks which embedded web objects are accessed frequently. Table 3 shows eight main web objects, number of access and related embedded web objects at Texas A&M University.

Table 3: Analysis of Web Objects

| Path | Access | E/O |
|---|---|---|
| /main.html | 1781 | 5 |
| /people/faculty | 351 | 5 |
| /academics/courses | 138 | 5 |
| /people/students | 113 | 4 |
| /academics/graduate | 105 | 5 |
| /department/images | 78 | 16 |
| /research/interests | 61 | 3 |
| /search | 55 | 5 |

**Step 2: Number of Web Processes**

Figure 4 shows the number of web processes of web server. The X axis shows the time of the simulation, while the Y axis shows the number of web processes in web server. Figure 4 shows average number of web processes in one hour. We compare our ConWebProc scheme and two static schemes including Static-2 and Static-4. Static-2 scheme and our ConWebProc create two web processes for incoming web requests, while Static-4 creates four web processes. Even though our ConWebProc creates only two spare web processes for incoming requests, ConWebProc shows higher performance than Static-4. Based on the prediction of incoming requests, ConWebProc controls number of web processes.

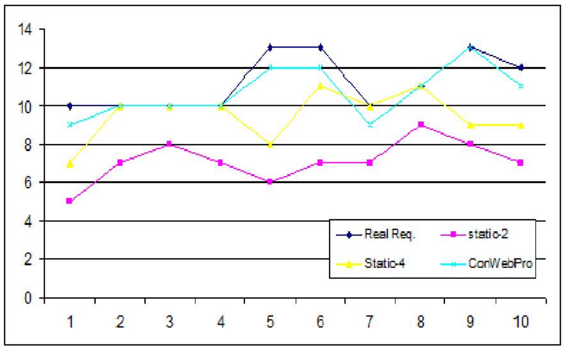

Figure 4. Number of Web Processes

**5. Conclusion**

In this paper, we applied ConWebPro to predict the number of incoming requests from web users based on the structure of requested web documents. Processes determine the performance of the web server. Inefficient process management creates too many processes, wasting the available memory of the web server, while a lack of processes delay transaction time due to the need to create new processes. To avoid these issues, ConWebPro decides in advance the proper number of web processes for incoming requests. In future work, we will research how web process management affects the overall performance of the web server.

**Acknowledgements:**

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2013R1A1A2063006)

**Corresponding Author:**
Gangman Yi
Department of Computer Science & Engineering
Gangneung-Wonju Nation University, Korea
E-mail: gangman@cs.gwnu.ac.kr

**References**

1. R. Sarukkai. Link prediction and path analysis using markov chains. Computer Networks, 2000.
2. B.D. Davison. Learning web request patterns. Web Dynamics: Adapting to Change in Content, Size, Topology and Use, pp. 435–460, 2004.
3. C. Bouras, A. Konidaris, D. Kostoulas. Predictive prefetching on the web and its potential impact in the wide area. World Wide Web: Internet and Web Information System, 2003.

4. T. Palpanas, A. Mendelzon. Web prefetching using partial match prediction. 4th Int'l Web Caching Workshop, 1999
5. A. Nanopoulos, D. Katsaros, Y. Manolopoulos. A data mining algorithm for generalized web prefetching. IEEE Transaction on Knowledge and Data Engineering, 2003
6. X. Chen, X. Zhang. A popularity-based prediction model fore web prefetching. of IEEE Computer, 2003.
7. Z. Ban, Z. Gu, Y. Jin. An online ppm prediction model for web prefetching. Web Information and Data Management, 2007.
8. Y. Chen, L. Qiu, W. Chen, L. Nguyen, R. H. Katz. Efficient and adaptive web replication using content clustering. IEEE Journal on Selected Areas in Communications, vol. 21, pp. 979–994, 2003.
9. G. Pallis, A. Vakali. Insight and perspectives for content delivery networks. Communications of the ACM, vol. 49, pp. 101–106, 2006.
10. N. Tuah, M. Kumar, and S. Venkatesh, "Resource-aware speculative prefetching in wireless networks. Wireless Networks, vol. 9, pp. 61–72, 2003.
11. S. Drakatos, N. Pissinou, K. Makki, C. Douligeris. A context-aware prefetching strategy for mobile computing environments. Int'l Conf' CMC, pp. 1109–1116, 2006.
12. P. Ferragina, A. Gulli. A personalized search engine based on web snippet hierarchical clustering. World Wide Web, 2005.
13. D. Cheng, R. Kannan, S. Vempala, G. Wang. A divide-and-merge methodology for clustering. SIG on Management of Data, 2005.
14. A. Serbinski, A. Abhari. Improving the delivery of multimedia embedded in web pages. Int'l Conf. on Multimedia, pp. 779–782, 2007.
15. E. Meneses, O. Rodriguez-Rojas. Using symbolic objects to cluster web documents. World Wide Web, 2006.
16. D. Kim, N. Adam, V. Alturi, M. Bieber, Y. Yesha. A clickstream-based collaborative filtering personalization model: Towards a better performans. WIDM, 2004.
17. L. Lu, M. Dunham, Y. Meng. Discovery of significant usage patterns from clusters of clickstream data. WebKDD, 2005.
18. F. Khalil, J. Li, H. Wang. Integrating markov model with clustering for predicting web page accesses. Australasian World Wide Web, 2007.
19. H. K. Lee, B. S. An, E. J. Kim. Adaptive prefetching scheme using web log mining in cluster-based web systems. pp. 903–910, 2009.

5/26/2014