Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 289 – 296

ISSN 1896-7094

An uncertain problem of caching dynamic content for web e-commerce

Waldemar Gil[1], Donat Orski[1]

[1] Institute of Information Science and Engineering
Technical University of Wroclaw
{Waldemar.Gil, Donat.Orski}@pwr.wroc.pl

Abstract. Improving the performance of the Web-based system is a crucial requirement, especially in e-commerce area. Many sites incorporate dynamic web pages to deliver customized content to their users. This is one of the most important motivations for improving the performance of the web-based dynamic content system. In many papers mining web logs are used for developing web caching and prefetching strategies. In this paper, we present benefit of this technique for boosting the performance of serving dynamic content. Generated documents are accommodated in a dedicated part of the cache, to avoid the drawback of incorrect replacement of requested documents. We exploit a method for variable-size content objects using an n-gram based prediction on future web request to improve the basic algorithm of buffering. The adaptation is reasonable where, because of an uncertainty, we could not perform the parameters of algorithm correction in an optimal way. Experimental results indicate improvement of web-access performance of dynamic content for this method.

1 Introduction

Caching has been viewed as an effective way to scale up Web sites and is generally used in all distributed information systems to reduce network traffic and improve response time for end users. Particularly, caching is important in the World Wide Web since the number of users on the web is increasing exponentially. Nowadays, users prefer to use customized and dynamic content. In contrast to static html pages, dynamic content is generated by web servers as answers to users' requests. The rate, at which page generation occurs, depends on the resource load placed upon the server by scripts running (such as ASP.NET, JSP, and PHP). These scripts require values of various parameters (e.g. article identifier) as well as necessary data from various sources (e.g. database servers) in order to create a html page, which may be returned to the user as an answer to their request. Although the dynamic content was very rare ten years ago, it has become the standard for web servers, especially in e-commerce area.

The advantages of dynamic pages comes with a trade-off, in that their demands more resources from the server. This is due to the requested answers being generated dynamically each time a request is made. A further consideration when dealing with dynamic content, arises when various similar requests are made (with the same *Uniform Resource Identifiers*, URI – based on RFC 2396), which are given very different answers (because having different parameters been specified – e.g. product id). Practically, these factors jointly lead to considerable load being placed in the host in question. This is a very interesting field of research [12, 13].

In order to increase the performance of generating dynamic web content, methods of reducing average request return time to a minimum, need to be well understood. Researchers and industry practitioners have described numerous ideas and approaches to solving these problems. Some papers discuss caching of entire files on web servers in dedicated buffers (proxy servers). Additionally, such papers address issues including pre-fetching of

predicted future requests [9]. In this paper, we discuss the concept of joining these two particular methods. We describe three methods of computing correction parameters in buffering algorithm.

2 Background

2.1 Caching in the Web

Most of the concepts for web caching ware taken from other aspects of computer and network design. For example operating systems have buffer caches for access to and from memory. Currently, web applications are becoming another popular area of caching.

Objects are buffered only as they are selected by an algorithm. Among the most popular algorithms is LRU (*Least-Recently-Used*). It is based on the assumption that the objects, which have been heavily requested in the past, will probably be requested again in the near future. An important disadvantage of this algorithm is that it fails to take into account the frequency which objects are requested with (as opposed to the sheer number of re-requests). This problem was solved by the import of expertise gleaned from database server methodology [11]. The solving of this problem led to the creation of LRU-K algorithm.

The next step in solving these problems consists in using object's size as a parameter in deciding whether an object is being captured in the buffer or not. A key value is attributed to each object. This key value relates directly to the time of generation of html page, and indirectly to the size of the object.

In order to decide which objects should be deleted, attention is paid to the key value, those objects which have the lowest key value are deleted first. The algorithm uses GDS (*Greedy Dual Size*) to achieve this [4, 8]. Just as was required in the case of LRU, an additional factor had to be considered, this factor again was the frequency of page that has been requested. This led to the creation of GDSF (*Greedy Dual Size Frequency*) [3].

2.2 Buffering of generated HTML objects

In the paper, we consider buffering of generated HTML objects, which depends on saving generated content in files on hard disk or in memory. If the next request concerns a page found in the buffer, then a file would be returned as an answer, instead of running the html-generating script.

The rules of computing the key value requires the introduction of a few notions. Let i be an enumeration of successive HTML object, differentiated by uniform resource identifiers with a full list of parameters, n – be a number of all differentiated objects ($i = 1, \ldots, n$), t_i – be a time of generation (period from the script's first being run, till the time of its ending) in milliseconds, u_i – be a size of newly generated object O_i in bytes, and y_i be a key value, being the rank of object O_i.

When object O_i is inserted into the buffer (in other words: the page O_i is generated by the scripts), then

$$y_i = \frac{t_i}{u_i} . \tag{1}$$

For the purpose of content being marked as disposable, the algorithm looks for the generated object of lowest key value y_i.

Let

$$h = \min_{i^*} y_{i^*} . \tag{2}$$

where i^* denotes only those objects which are in the buffer.

When the object O_i is being replaced (deleted from the buffer) by another page (which is inserted into the buffer) the key value is increased by the h value:

$$y_i = h + \frac{t_i}{u_i} . \tag{3}$$

In such cases, the replacement is necessary to delete a few objects from the buffer, the key value is increased by a sum of h value. Using this algorithm also requires a maximum value of buffer size in bytes U.

2.3 User requests

Let j be an index of successive user request, and m be a number of all requests (j = 1, …, m). Every request r_j concerns only one object: $r_j \in (1, \ldots, n)$.

When request r_j concerns an object O_i and object has a static form in the buffer the key value y_i is being increased:

$$y_i := y_i + \frac{t_i}{u_i} . \tag{4}$$

When requested object has not a static form, the HTML page of object O_i was generated by executing appropiate program or script. If the buffer's free space is larger or equal to the size of the HTML page Equation 1 is used to compute the key value y_i. In opposite case Equation 3 should be used.

We observe that this algorithm has some disadvantages. Sometimes removing the object followed the step before incoming the next request. The basic problem is lack of information about frequency of object's requests in the long period. One of suggested solutions consists in introducing the key value correction, based on prediction of future request and data mining methods.

2.4 Correction of Buffering Algorithm

Correction of buffering algorithm was presented in [9]. Correction of algorithm relies on taking into consideration W_i measure when computing key value y_i. Equation 1 needs to be replaced by Equation 5:

$$y_i = (1 + W_i)\frac{t_i}{u_i} . \tag{5}$$

Similarly, Equation 3 needs to be replaced by Equation 6:

$$y_i = h + (1 + W_i)\frac{t_i}{u_i} . \tag{6}$$

Scaling the value of future frequency of requests can proceed independently from operation of algorithm. It results from the specificity of data mining or other methods and availability of source data. Computing W_i can proceed with an hour, day or extra period.

In [9] was presented method of computing W_i values based on sequence mining. Main problem described in [9] was demand of high computational power. We can prepare other methods of computing parameters of caching algorithm which will require less time for computing W_i values. On the one hand we can decrease number of seconds required for computing parameters, but on the other hand there is a question about time required for generating objects and running the algorithm with correction. Can we save more seconds for

computing parameters than we can lose for generating objects? That is a good question for the web servers' owners. As an answer we need to prepare a performance analysis of methods of computing parameters and implement two correction parameters ratios.

2.5 Correction Parameters Ratios

Manner of introduction of key value correction y_i may not be optimal because of uncertainty. For example, if we use W_i value multiplied by two or three, can we expect better results? For that reason we propose to change this model to more general.
Let p be the linear correction parameters ratio and Equation 5 needs to be replaced by Equation 7:

$$y_i=(1+pW_i)\frac{t_i}{u_i} \quad . \tag{7}$$

Let q be the exponential correction parameters ratio and Equation 5 needs to be replaced by Equation 8:

$$y_i=(1+W_i^q)\frac{t_i}{u_i} \quad . \tag{8}$$

Two ratios can be used simultaneously:

$$y_i=(1+pW_i^q)\frac{t_i}{u_i} \quad . \tag{9}$$

We can use the process of adaptation to assign the best values of two ratios and to improve the performance of web-based dynamic system.

3 Methods of Computing Parameters

3.1 Data Source

For computing W_i value it is important to choose the correct data source. Natural choice [9] will be set of history of users' request – web server log. Each line of log includes full description of request (e.g. IP address or the name of host, the date and time, the method to be applied to the resource, the resource identifier – URI, protocol version, and so on). Basing on the resource identifier it is possible to categorize of requested objects.

Information about requests is written down according to a sequence of realization. Log can be the main source of information about users' visits and preferences. Based on this sequence it is possible to train a path-based model for prediction of future requests.

3.2 Sequence Mining and Frequency of Requests

The web systems analysis with use of data mining methods and tools is often used in many areas. The most popular are classification, discovering association rules and prediction of future request [5, 6, 7].

The first method of computing parameters is analysis of visit. In order to carry it out we can use sequence mining method because information on sequence of requests is very important [1, 2]. We build an n-gram prediction model based on web server logs. This model is based on object-occurrence frequency. At first, we

divide each log into a set of sub-logs containing requests of a single visit. Afterwards, the algorithm scans all sub-logs, counting occurrence frequencies of objects' requests of the next l click after the sub-log in all visits.

The result of counting is the conditional probability of future objects' requests [13]. As an example, consider a sequence of requests in server log: {A, B, A, C, A, C, B}. The two-grams with the predicted object's shown in Table 1, and their conditional probabilities.

Table 1 . An example of n-gram model

Two-gram	Conditional probability
A, B	{<A, 100%>}
A, C	{<A, 50%>, <B, 50%>}
B, A	{<C, 100%>}
C, A	{<C, 100%>}
C, B	{<A, 100%>}

By default n-gram prediction model concerns only one visit, but normally there co-exist a number of visits on a web server.

Different visits will give different predictions for future objects. Because our prediction of an object comes with a probability of its request, we can join these predictions to calculate the future occurrence frequency of an object.

Let S_k denote single user's visit, where k is a number of all visits, k = 1, …, K, and $P^{\mathrm{I}}_{i,k}$ be a conditional probability predicted by a visit S_k for object O_i. If object O_i is not predicted by visit S_k, then $P^{\mathrm{I}}_{i,k} = 0$. We assume that all the visits on web server are independent and let W_i^{I} be the future frequency of requests to object O_i:

$$W_{iI} = \sum_j P^I_{i,j} \ . \tag{10}$$

As an example, consider a prediction for two visits and six objects, shown in Table 2.

Table 2. An example of prediction $P^{\mathrm{I}}_{i,k}$ for three visits

O_i	O_1	O_2	O_3	O_4	O_5
$P^{\mathrm{I}}_{i,1}$	0.65	0.55	0.00	0.24	0.00
$P^{\mathrm{I}}_{i,2}$	0.25	0.37	0.82	0.00	0.00
$P^{\mathrm{I}}_{i,3}$	0.00	0.29	0.05	0.31	0.45

The probability predicted for all objects are equal to:
$W_1^{\mathrm{I}} = 0.65 + 0.25 = 0.90$
$W_2^{\mathrm{I}} = 0.55 + 0.37 + 0.29 = 1.21$
$W_3^{\mathrm{I}} = 0.82 + 0.05 = 0.87$
$W_4^{\mathrm{I}} = 0.24 + 0.31 = 0.55$
$W_5^{\mathrm{I}} = 0.45 = 0.45$

3.3 Counting Requests within Visits

Second method of computing correction parameters is counting requests within visits. In this method we must also divide each log into a set of sub-logs containing requests of single visit, but we do not need to use full sequence mining algorithm [10].

Let S_k denote single user's visit, where k is a number of all visits, $k = 1, \ldots, K$ as in first method of computing parameters. In this method let $P^{II}_{i,k}$ be the probability of request object O_i within visits S_k.
Let W_i^{II} be:

$$W_{i}^{II} = \sum_{j} P^{II}_{i,j} \ . \tag{11}$$

Computing parameters W_i^{II} can be done without sequence part of mining algorithm.

3.4 Counting Requests of Categorized Objects

Sequence discovery and analysis of visit need using web mining methods. Those methods require high computational power. In case of lack of power we will propose methods without web mining. One of them consists in counting requests of categorized objects.

Let l be an enumerator of category of objects, v be a number of all categories ($l = 1, \ldots, v$). For each object O_i, which can be requested by user, we can assign a category $c_i \in (1, \ldots, v)$.

Third method of computing correction parameters W_i^{III} consists in accounting of requests of object O_i relatively to a number of requests of objects in category c_i.

4 Results of the Experiments

In experiments we did compare the time required to generate pages (in seconds), which is defined as a number of request, which lead to pages, being retrieved from the buffer, against the sum of all requests made. Our results were obtained from a set of 20,000 users' requests. Requests were made to approximately 1,000 different dynamic pages.

We described typical workload and its popularity given by well known Zipf-like distribution. Workload file size distribution was described in [14] and was used in [10]. An example of results with three methods of correction and without correction (only GDSF) of buffering algorithm was shown in Fig. 1. Correction of algorithm was made every 200 requests.

Implemented correction lead to an improvement in principal parameter, that has been measured: time required decreased. This algorithm (all version with correction) is therefore preferable in situations, in which the main criterion is the time required for processing.

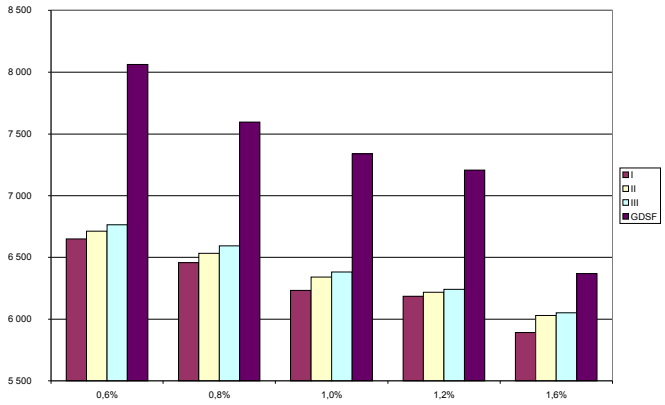

Fig. 1. Time versus cache size with three methods of correction and without correction

Method I (based on sequence mining) of correction parameters is the best in all shown in Fig. 1 variants. Method without correction (only the basic GDSF algorithm) is the worst. All time intervals was measured on PC computer with Pentium IV 3 GHz.

We can compare results of methods of correction with different linear and exponential ratios.

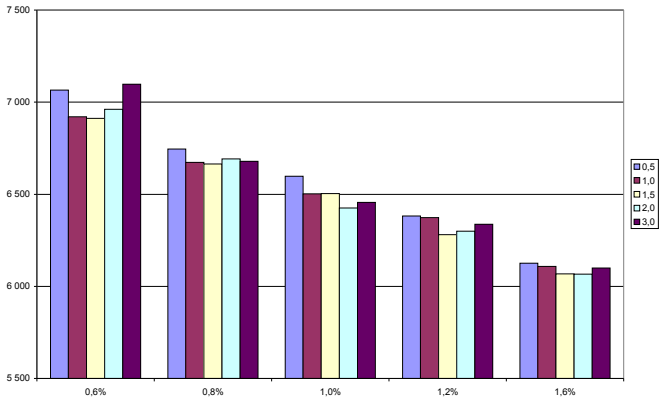

Fig. 2. Time versus cache size and linear ratio

One can clearly see that the time value is minimum with linear ratio between 1.0 and 2.0. It also depends on cache size.

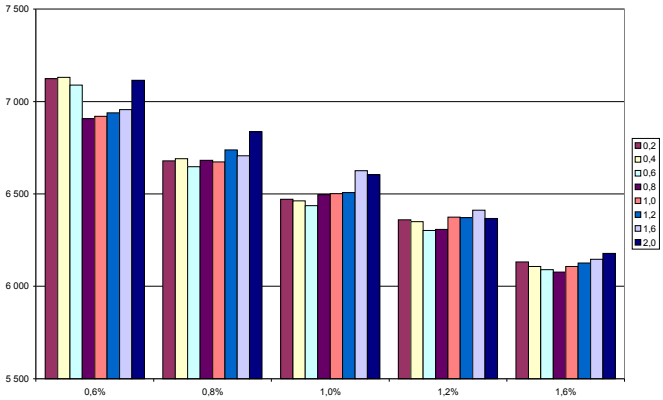

Fig. 3. Time versus cache size and exponential ratio

Generally, for exponential ratio, the best results one can obtain while ratio is below 1.0.

While we using two ratios simultaneously, one can observe the best results. As an example, consider results with and without ratios, shown in Table 3.

Table 3. An example of results with ratios and with GDSF algorithm only

p	*q*	Sum of time	%
GDSF		8.062	100.0
1.0	1.0	7.134	88,5
1.0	0.8	7.089	87.9
1.5	0.8	6.905	85.6

5 Conclusion

In this paper we have discussed analysis of three methods of computing parameters in caching of dynamic web content. Additional correction of the algorithm under discussion has led to obvious improvement obtained in the most recent results. Additional ratios of correction parameters have a positive impact on system performance, as seen by the end user.

We are interested in the field of developing techniques for e-commerce area to divide pages into regions, whereby some parts of page are 'earmarked' as candidate areas to accept buffered content. Such techniques offer great potential for obtaining improved performance from any given size of buffer.

References

1. Agrawal R., Imielinski T., Swami A., *Mining association rules between sets of items in large databases,* Proceedings of the 1993 SIGMOD Conference, Washington 1993, pp. 207-216.
2. Agrawal R., Srikant R., *Fast algorithms for mining association rules,* Proceeding of 20th International Conference on Very Large Data Bases, VLDB, 1994, pp. 487-499.
3. Arlitt M., Friedrich R., Cherkasova L., DIlley J., Jin T., *Evaluation content management techniques for web proxy caches,* HP Technical report, Palo Alto, April 1999.
4. Aggarwal C., Wolf J. L., Yu P. S., *Caching on the World Wide Web, IEEE Transactions on Knowledge and Data Engineering,* v. **11**, 1999, pp. 94-107.
5. Borzemski L., Data mining dla Internetu, ZN Pol. Śl. Studia Informatica Vol. **24**, No 2A (53), Gliwice, Poland 2003, pp. 109-118.
6. Berendt B., Hotho A., Mladenic D., Someren M., Spiliopoulou M., STumme G., *A Roadmap for Web Mining: From Web to Semantic Web,* Proceedings of the 1st European Web Mining Forum, Dubrownik, Croatia, 2004, pp. 1-22.
7. Cichosz P., *Systemy uczące się,* Wydawnictwo Naukowo-Techniczne, Warszawa, Poland 2000.
8. Cao P., Irani S., *Cost-aware WWW Proxy caching algorithms,* Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, CA, December 1997.
9. Gil W., *Zastosowanie eksploracji danych do zwiększenia wydajności generowania dynamicznej treści WWW,* Wydawnictwo Komunikacji I Łączności, Warszawa, 2005, pp. 425-434.
10. Gil W., *Metody wyznaczania parametrów korekty algorytmu buforowania dynamicznej treści WWW,* Inżynieria Wiedzy i Systemy Ekspertowe, Wrocław, 2006, pp. 115-124.
11. O'Neil E. J., O'Neil P. E., Weikum G., *The LRU-K page replacement algorithm for database disk buffering,* Proceeding of the 1993 ACM SIGMOD International Conference on Management od Data, May 1993, pp. 297-306.
12. Vassilakis C., Lepouras G., *Controlled Caching of Dynamic WWW Pages,* Database and Expert Application: 13th International Conference, DEXA 2002, LNCS 2453, Aix-en-Provence, France, September 2002, pp. 9-18.
13. Yang Q., Hang H., Li I., Lu Y., *Mining Web Logs to Improve Web Caching and Prefetching,* Web Intelligence: Research and Development, Maebashi City, Japan 2001, pp. 483-492.
14. *SPEC benchmark for evaluating the performance of World Wide Web Servers,* `http://www.spec.org/web2005`