Using Database Technology to Improve Performance of Web Proxy Servers*

Kai Cheng, Yahiko Kambayashi
Graduate School of Informatics
Kyoto University
Sakyo Kyoto 606-8501, Japan
{chengk, yahiko}@db.soc.i.kyoto-u.ac.jp

Mukesh Mohania
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008, U.S.A.
mohania@cs.wmich.edu

Abstract

In this paper, we propose to use database technology to improve performance of web proxy servers. We view a cache at a proxy server as a web warehouse with data organized in a hierarchical model, which consists of physical pages, logical pages and topics, similar to data organization in database systems. By defining priority on each abstraction level, the cache manager can make replacement decisions based on topics, logical pages and finally physical pages hierarchically. We verify the proposed approach by developing a content-aware caching scheme, namely LRU-SP+. We evaluate our scheme in terms of hit ratio (HR) and profit ratio (PR) which differentiate topics of different pages. The results show that LRU-SP+ generally performs 30% better than a content-blind scheme.

1 Introduction

A proxy server is a special hypertext server that acts as an intermediary between web servers and clients, providing access to the web for people who can only access the Internet through a firewall [8]. While caching has been extensively used to improve performance of web proxy servers, state-of-the-art caching schemes do not perform so well as desired, with a upper bound of 30%-50% in hit ratios [1]. This is partially because most web clients also have local caches which absorb most shortly recurring accesses. It is reasonable to have proxy servers primarily to cache *longterm popular* contents instead of *temporarily popular* ones. Longterm popular contents should then be made explicitly accessible like a large web warehouse to maximize the utilization. This has been beyond the capability of the state-of-the-art caching schemes where all web documents are treated uniformly as physical pages with contents transparent to users as well as cache manager. Basically, users' access patterns are independent of whether there is a cache and what are currently available in the cache.

We have noticed the importance of exploiting semantic information and user preferences for cache management. In [3], we have proposed a constructive approach for design and analysis of advanced cache replacement policies, in which a cache agent consists of a central cache and multiple unit caches. Contents of a web cache are managed in different unit caches according to *classification rules*, a certain logical rules possibly based on semantics of documents. For example, if a document D is from Japan and the content is about baseball, the type is picture and the size is bigger than 24KB, then keep D in unit $\#u$. In [5], we discussed for the first time the issue of content management for web caching, and proposed a multicache-based architecture to meet this needs. Despite these efforts, however, the semantic information in these schemes only plays a minor role, since only classification rules use semantic information while neither central caches or unit caches (or subcaches) are virtually content-blind. All this previous work, however, stimulated us to explore more advanced, database-like approach for content management and cache enhancement.

In this paper, we present a new scheme for proxy caching that empowers users and cache manager with *content-aware* capability. First, based on database technologies, we develop a hierarchical model for web data. By defining priorities over topics, logical pages and physical pages respectively, the cache manager can make replacement decisions hierarchically based on the topics, logical pages and physical pages. As a result, the cache can always keep the

*In *Proceedings of the Fourth International Workshop on the Web and Databases (WebDb'2001),* May 2001. Santa Barbara CA

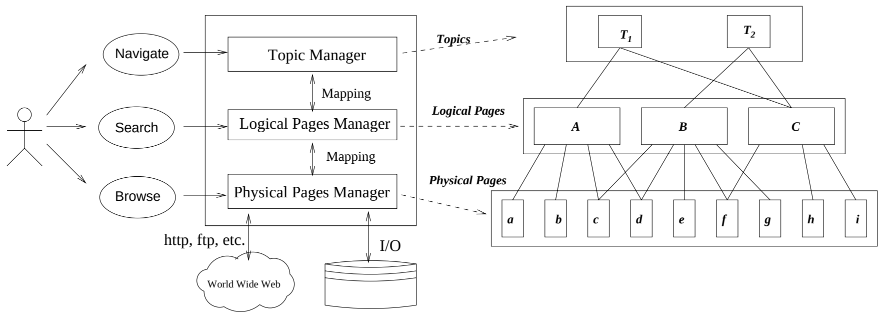

Figure 1: The hierarchical model for web data

most valuable, popular as well as fresh contents. We verify the latter by developing a content-aware caching scheme, namely LRU-SP+. We evaluate our scheme in terms of hit ratio (HR) and profit ratio (PR) by taking into account the differences in popularity of the topics of a page. The results show that LRU-SP+ generally performs 30% better than the content-blind scheme.

The organization of this paper is as follows. In Section 2, we describe a hierarchical model for web data and the corresponding system architecture to facilitate content management for proxy caches. Section 3 presents a content-aware replacement algorithm, LRU-SP+. Section 4 discusses and experimentally evaluate the performance of LRU-SP+.

2 A Hierarchical Model for Web Data

In this section, we introduce a hierarchical model for web data to facilitate content-aware cache management and content utilization. On the one hand, to facilitate cache management, it should be easy to locate the topics that a document is most relevant to and to decide the popularity of them. From the point of view of content utilization, it is desirable to organize the web data in terms of topics or categories. In addition, hyperlinks are precious information for determining the information structure. Many documents might be of no use if some linked documents lost. The data model for content-aware caching is based on a hierarchical structure that supports different abstraction level of web data.

2.1 The Hierarchical Structure of Web Data

Figure 1 shows the hierarchical structure of web data, where physical pages are organized to be logical pages, logical pages are then classified into topics. The three levels of abstraction are described as follows:

1. *Physical Page*, single physical file, which can be either a HTML document or an embedded media file (Figure 1). All web caching policies so far developed deal with physical pages (often called web objects)

2. *Logical Page*, a set of physical pages with one *main page* and several directly linked/embedded physical pages. A HTML physical page together with its all embedded media components form a logical page. Logical page is suitable information unit for searching and classifying.

3. *Topic*, a set of logical pages relevant to a given topic. Logical pages are categorized in terms of various features of them such as keywords, home server, language and etc.

When physical pages were retrieved, a logical page generator will first generate one or more logical pages for these physical pages. Then a topic classifier is used to determine which topic(s) a logical page should belong to. Conversely, to purge a document from the cache, a candidate topic must be selected at first, then a logical page in this topic, and finally one or more physical pages within this logical page will sequentially chosen. The content-aware cache management scheme to be developed later in this paper is built upon the two basic operations described above, in addition with definition of *priority orders* or *replacement policies* for all abstraction levels.

2.2 An Architecture for Implementing the Hierarchical Data Management

The data abstraction and management lead to a hierarchical system architecture as shown in Figure 1. There are three managers responsible for managing the aforementioned three-level web data: physical page manager (PPM), logical page manager(LPM) and topic manager(TPM) respectively.

Physical page manager (PPM) acts as a simple cache manager, servicing the incoming requests, maintains the cached physical documents by a priority queue. At the lowest level, such management is restricted or supervised by LPM and TPM. *Logical page manager (LPM)* is responsible for generating and maintaining the set of logical pages. Links are important metadata for hypermedia documents. Logical page generator analyzes the link structure for a set of linked documents, grouping those with close linkage into a logical page. Another task of LPM is to maintain logical pages, defining priority among different logical pages for cache replacement. The policy for logical page replacements is based on the reference history, as well as the relevance between the page and the topic cluster. *Topic manager (TPM)* is at the highest abstraction level that classifies logical pages into predefined topics, managing the priority between topics for cache replacement, and providing a directory of cache content for people to navigate.

When it is necessary to replace some documents to make space for more potential ones, the cache manager begins to choose a topic that is the least popular/important ones among all topics, then choosing logical pages of least priority under these topics, and finally, physical pages are selected among the physical pages that belong to the candidate logical pages.

3 Cache Replacement Policy Based on the Hierarchical Model

Based on the hierarchical data model for content-aware cache, in this section, we present the content-aware cache management scheme. The basic idea is to make replacement decisions beginning with choosing a *topic* with least priority, then down toward the lower level decisions. We will first describe this multi-layered cache replacement policies in general. Then, we give a concrete algorithm, namely, LRU-SP+, which has been proved a successful cache scheme through experimental evaluations to be described in the next section.

3.1 Policy for Cache Replacements Based on Hierarchical Content Management

The first round choice for a replacement begins from the topics. The least popular topic, e.g B will be chosen to continue the next round choice. The priority of each topic are based on a pre-specified order in terms of group preference or other control policies. For example, within a company, it is reasonable to give leisure or even sex topics lower priorities to have more cache space for topics on work and business. This mechanism provides us with a flexible topic-based control over the performance of proxy caches.

After B has been decided as the candidate topic, from which we can replace some pages to make space, the next round choice is a little complicated since choosing a logical page within the selected topic requires a tradeoff between the a page's relevance to the topic and the popularity shown so far. Now, the decision left is straightforward, since we just need a traditional caching policy applicable to most the physical page context. As shown in Figure 2, the scope for cache replacement is restricted from the dashed circle to the dash-dotted circle, that is, $\{D_1\{E_1, E_2\}, D_2, D_3\{E_3, E_4, E_5\}\}$ Caching policies that can be used in this case include LRU(least Recently Used), LFU(Least Frequently Used), Size-Adjusted LRU[2] and any good algorithms developed so far. We have proposed a novel replacement algorithm, namely LRU-SP [4], which has proved well suited for caching physical pages. In the following, we will review the LRU-SP algorithm.

3.2 LRU-SP+: A Content-Aware Extension to LRU-SP

LRU-SP (Size-adjusted and Popularity-aware LRU) is an extension to LRU (Least Recently Used) proposed by K. Cheng et al in [4]. The basic idea behind is that if one hit saves a unit of time and retrieval cost, then more hits should reasonably save more units of time and cost. Thus, the benefit/size function of document i with RF_i references should be: $RF_i \cdot (1/\Delta T_{it})/S_i$, Therefore, to choose an document with least benefit, we should re-index all documents in cache in an increasing order on values of $(S_i \cdot \Delta T_{it})/RF_i$ (instead of $S_i \cdot \Delta T_{it}$), then, greedily picked the highest index objects one by one and purge from the cache until sufficient space being made.

$$\frac{S_1 \cdot \Delta T_{1t}}{RF_1} \leq \frac{S_2 \cdot \Delta T_{2t}}{RF_2} \leq \cdots \leq \frac{S_k \cdot \Delta T_{kt}}{RF_k}$$

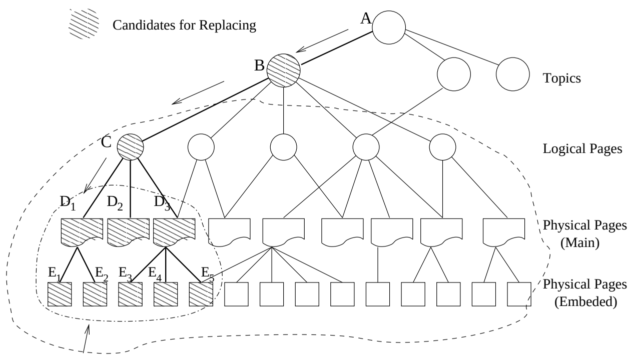

Figure 2: Cache replacements from topics down towards physical pages

By employing LRU-SP as the physical page manager in a content-aware cache management scheme, we get a content-aware LRU-SP, we call it *LRU-SP+*.

4 Experimentations

Content-aware cache management integrates several key techniques in a singe scheme. The performance of content-aware proxy caching can be evaluated from various aspects. First, it is a hard work to find relevant and high quality information from the Web sites all over the world. Content-aware proxy caching system offers various facilities for resources sharing among like-minded users. Second, our approach provides higher level performance tuning and control for system administrators, who can now play around tangible information units such as topics, logical page. Finally, cost saving in time and network bandwidth is the basic feature of most caching schemes.

In this paper, we evaluate and compare the efficiency of our scheme with LRV. LRV is a new caching scheme especially designed for Web caching [9]. The designers have developed an elaborate function to handle various characteristics of Web objects. An efficient content management scheme is given to LRV, which also classifies objects into a few groups according to their access frequency. Objects in the first group are maintained using a unit caching policy SIZE, whereas the rest groups are FIFO lists. Final decisions are made by a central cache which is based on LRV-function.

4.1 Experiment Design

As document clustering/categorization and link structure extraction are well researched in information retrieval (IR) as well as the World Wide Web, in this work, we focus on evaluating the effect of differentiating value of topics. To do this, we define *significance factor* for each topic. A significance factor is a real number between 0 and 2. The most significant topic is weighted 2. The default is 1.

Instead of using byte hit ratio (BHR), which measure the efficiency of caching in terms of the data size that has been served by cache, we use another weighted hit ratio, namely *profit ratios (PR)*. Let $\omega_i \in [0, 2]$ be the significance factor, d_i be the document corresponding to the $i'th$ request and N be the number of requests seen by the cache. then

$$PR = \frac{\sum_{i=1}^{i=N} \omega_i \times y_i}{\sum_i \omega_i}, \quad y_i = \begin{cases} 1 & \text{if } d_i \text{ in cache} \\ 0 & \text{otherwise} \end{cases}$$

The dataset we used to drive our simulator are access logs as well as the corresponding Web data collected from the Squid proxy server in our laboratory. The logs keep all of 873,824 requests for total 23.6 GB Web data with the maximum hit ratio 0.251 and byte hit ratio 0.098. The Web data has 21.3 GB in size. About 75% of the data have never been used since they were retrieved for the first time. Even when considering the margin errors. Thus, it is a significant progress if we can make best use of these web data.

In Table 1, we list the major topics to be considered according to the status of the laboratory. The priority between thee topics is assigned based on significance and popularity.

Topics	Description	Priority
EDU	Distance Education	3 (1.5)
HCI	Human Machine Interface	1 (2.0)
GIS	Geographical Information System	3(1.5)
LOG	Logic Design	3 (1.5)
PRG	Programming	1 (2.0)
LEI	Leisure and Recreation	3 (1.0)
OTR	Other	5 (0.5)

Table 1: Topics and their priority(topic significance)

4.2 Results and Analysis

The experimental results under different datasets are shown in Figure 3. The subfigures depict the hit ratios and profit ratios achieved when using trace dataset KAMB as input. Both hit ratio in terms of how many requests are satisfied by cache, and profit ratio, a weighted version of hit ratio by considering the significance factor of topics, performs better than the baseline algorithm LRV.

First, LRU-SP+ performs about 20% better than LRV in terms of hit ratio as shown in Figure 3(left). This is because in this laboratory, the selected topics are the major concerns of students and staff. By giving higher priority to these topics, we can indeed keep almost all popular contents. The documents that are not relevant to the these topics are rarely accessed, which are less likely cached due to the lower priority and even cached, they may quickly be replaced. The topic-based priority guarantees popular contents being cached long enough, whereas less popular ones will not occupy cache space. Second, in terms of PRs, LRU-SP+ outperforms over LRV by a factor of 30% or so as shown in Figure 3(right). This is reasonable because in our significance factor assignment, contents with higher priority are assigned a relatively high profit. This factor further enlarge the benefits obtained from the hit ratio.

The performance of this algorithm relies on the topic categorization algorithm ($l2T(\cdot)$). There is a long list of studies on automatic text categorization, among which, SVM (Support Vector Machine) has been regarded as a most powerful one for its fully automatic property eliminating the need for manual parameter tuning[6, 7]. We have adopted this technique. For the complexity in implementation, at present we just manually set topic categories and all topics have only a flat structure without defining a hierarchical structure for all possible topics. To scale the scheme, it needs well-established automatic clustering algorithms. However, even a simple form as the implemented one, it is interesting and useful for companies or organizations that wish their system resources be used in business-related web accesses. It is also effective since a well-defined company or organization can always provide a definite description of their interest.

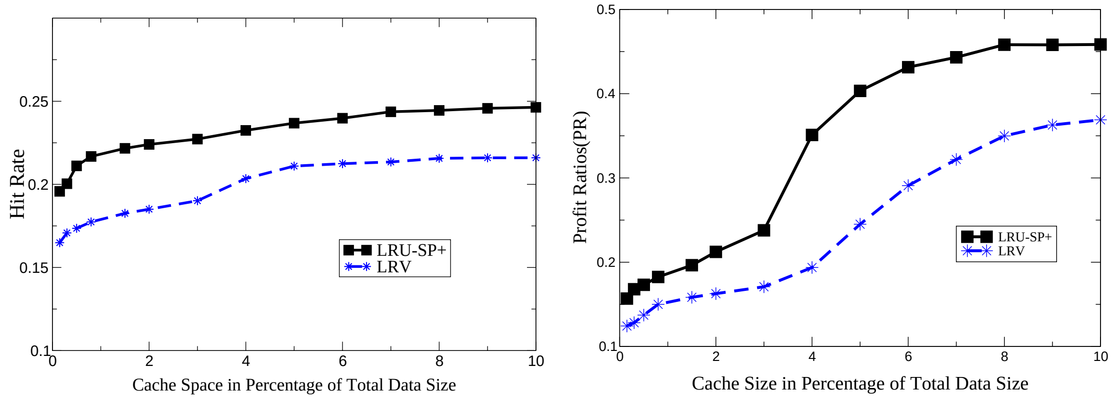

Figure 3: Experimental results

5 Concluding Remarks

Large proxy caches are increasingly important for efficient utilization of web contents and network resources. In this paper, we have proposed a framework of a proxy caching where both users and cache manager are aware about the content of web documents. It is the first time, to the best of our knowledge, that a proxy cache is treated as a shared information repository, rather than simply a collection of physical data. A hierarchical data model was developed to exploit the cache contents and their semantic information. Based on this model, we designed and implemented LRU-SP+, a content-aware algorithm for web proxy caching. The experimental results show that the proposed approach is a promising way to solve the problems caused by the exponential growth of the web size and Internet traffic. For future work, some more experiments are required and more work should be done to deal with the scalability issues of the proposed approach, that is, how to scale to large cache with many topics and a number of user communities. It is also necessary to develop facilities for using cache as a web warehouse to enable more active use of cache contents.

References

[1] M. Abrams, C. R. Standridge, G. Abdulla, S. Wililams, and E. A. Fox. Caching Proxies: Limitations and Potentials. In *Proceedings of the Fourth International WWW Conference*, 1995.

[2] C. Aggarwal, J. L. Wolf, and P. S. Yu. Caching on the World Wide Web. *IEEE transactions on knowledge and data engineering*, 11(1), 1999.

[3] K. Cheng and Y. Kambayashi. Advanced Replacement Policies for WWW Caching. In *Proceedings of 1st International Conference on Web Age Information Management(WAIM'2000), LNCS 1846*, pages 239–244. Springer-Verlag, June 2000.

[4] K. Cheng and Y. Kambayashi. LRU-SP: A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Caching. In *Proceedings of 24th International Computer Software and Applications Conference (Compsac'00)*, pages 48–53, 2000.

[5] K. Cheng and Y. Kambayashi. Multicache-based Content Management for Web Caching. In *Proceedings of 1st International Web Information Systems Engineering(WISE'00)*, pages 42–49, June 2000.

[6] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *European Conf. Mach. Learning, ECML98*, Apr. 1998. TR 23, Univ. Dortmund, Lehrstuhl Informatik III.

[7] J. T.-Y. Kwok. Automated Text Categorization Using Support Vector Machine. In *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, pages 347–351, Kitakyushu, Japan, October 1998.

[8] A. Luotonen. World-Wide Web Proxies. In *Proceedings of the First International World Wide Web Conference*, Geneva (Switzerland), May 1994.

[9] L. Rizzo and L. Vicisano. Replacement Policies for a Proxy Cache. Technical report rn/98/13, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK, 1998. http://www.iet.unipi.it/ luigi/ caching.ps.gz.