Knowledge and Information Systems (2002) 4: 202–218

Knowledge and Information Systems

Enhanced Proxy Caching with Content Management

Kai Cheng, Yahiko Kambayashi

Department of Social Informatics, Graduate School of Informatics, Kyoto University, Kyoto, Japan

Abstract. In this paper, we propose a novel approach to enhancing web proxy caching, an approach that integrates content management with performance tuning techniques. We first develop a hierarchical model for management of web data, which consists of physical pages, logical pages and topics corresponding to different abstraction levels. Content management based on this model enables active utilization of the cached web contents. By defining priority on each abstraction level, the cache manager can make replacement decisions on topics, logical pages, and physical pages hierarchically. As a result, a cache can always keep the most relevant, popular, and high-quality content. To verify the proposed approach, we have designed a content-aware replacement algorithm, LRU-SP+. We evaluate the algorithm through preliminary experiments. The results show that content management can achieve 30% improvement of caching performance in terms of hit ratios and profit ratios (considering significance of topics) compared to content-blind schemes.

Keywords: Content management; Information retrieval; Performance; Proxy caching; Web database

1. Introduction

Caching 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 data and the number of users on the web are increasing exponentially, far outpacing the increase of network bandwidth. Recent study shows that data on the web in 1999 has increased four times (partially due to creating mirror web pages and automatic generation of data) compared with the previous year (Lawrence and Giles, 1999), whereas the network bandwidth increased only twice (Nielsen, 1998). To reduce the amount of traffic on the

Received 19 Mar 2001
Revised 17 Apr 2001
Accepted 25 May 2001

net, to reduce server load and to speed up data access, popular and often-used documents should be cached at local machines (Aggarwal et al., 1999; Barish and Obraczka, 2000).

Most concepts for web caching are taken from other aspects of computer and network design. Modern CPUs have caches for access to and from memory. Modern operating systems have buffer caches for access to/from disks. Distributed file systems have caches for data access between clients and servers. However, web caching, particularly shared proxy caching, differs from its predecessors in several ways. First, web caches, particularly the shared proxy caches, are usually very large in size, up to tens or hundreds of gigabytes. This is in contrast to the caches used in operating systems where cache size is small and data is kept for a relatively short period of time. Since proxy cache can be large and is a shared information repository, it can be seen as a warehouse of web documents, where people can share each other's findings and efforts. However, a cache is conventionally used just in a passive way where access patterns of users are independent of the cache. The web is highly open, dynamic, and less structured in nature, and therefore it is laborious for people to locate relevant and high-quality information. To improve the utilization of cache contents and to facilitate information sharing between like-minded people, the cache contents need to be well organized.

Secondly, and perhaps most importantly, the users of a web cache are people with special information needs. Information needs represent the contents that users would like to access. For example, users who want to learn more about research into 'usability' would access documents pertaining to something like human–computer interfaces, user-centered design, etc. Users with similar information needs tend to access a similar set of web documents, while users with completely different information needs would rarely share information with each other. Therefore, information needs based on contents of web documents are a key factor in web-caching systems.

However, caching schemes developed so far are primarily based on the past usage of data, such as how often the data has been accessed, to predict the access pattern in the future (Cao and Irani, 1997; Williams et al., 1997; Wooster and Abrams, 1997). To do this, however, each object should be kept in cache for a period of time before one can tell whether it should continue to stay. For example, in an LRU (least recently used) caching scheme, each object may stay in cache until it becomes the least recently used object. Usage-based schemes are dominant in traditional caching because additional information other than usage is not directly available to the cache managers and the time scales allowed in those systems are so narrow that it is unrealistic to implement complicated algorithms to analyze the system's behavior.

We have noticed the importance of exploiting other factors, especially semantic information, in cache management. In Cheng and Kambayashi (2000a), we 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 (retrieved web documents) are managed in different unit caches according to a set of *classification rules* based on semantic information in documents. For example, if a document D is from Japan and the content is about baseball, type is picture and size is bigger than 24 kB, then keep D in unit $\#u$. In Cheng and Kambayashi (2000c), we discussed the issue of content management for web caching, and proposed a multicache-based architecture to meet this need. This is the first study that suggests integrating content management with performance-tuning techniques. Despite these efforts, however,

semantic information in these schemes play a small role, and are only used in classification rules, while neither central caches or unit caches (or subcaches) are virtually content-blind.

In this paper, we present a new approach to enhancing proxy caching based on techniques of content management. We first propose a hierarchical model for web contents, consisting of physical pages, logical pages and topics. We then develop searching and topic navigation facilities for users to quickly find what they would be interested in. Furthermore, by defining priority on topics, logical pages, and physical pages respectively, the cache manager can make replacement decisions by choosing for replacement the candidate topics, logical pages, and physical pages hierarchically. As a result, the cache always keeps the most relevant, popular as well as high-quality content. We verify the proposed scheme by developing a content-aware caching algorithm, namely LRU-SP+. We evaluate our scheme in terms of hit ratio (HR) and profit ratio (PR) taking into account the differences in popularity of the topics of pages. The results show that LRU-SP+ generally performs 30% better than the content-blind scheme.

The remainder of this paper is organized as follows. Section 2 describes the hierarchical model and system architecture to facilitate the management and utilization of web contents. Section 3, discusses active access to cached web contents through indexing and categorization. In Section 4, we present a content-aware replacement algorithm for proxy caching, namely LRU-SP+. In Section 5, we discuss the advantages and disadvantages of the proposed approach. We experimentally evaluate LRU-SP+, comparing it with LRU-SP and another baseline algorithm, LRV (least relative value). Section 6 briefly reviews related work. Section 7 concludes this paper and describes several directions of future work.

2. A Hierarchical Model for Content Management

Conventionally, contents of a cache are primarily managed as physical data using very simple data structures such as priority queue with or without a hash table. Web data, however, are multimedia hypertext, and the hyperlink structure of web documents determine how users will navigate the information space. To exploit this feature in cache management, web data should be well organized according to a well-designed model. Borrowing the basic data-modeling concepts from database systems, in this section we introduce a hierarchical model for web data to facilitate content-aware cache management and content utilization. The major goal is to support different abstraction of web data so that both cache user and manager can efficiently utilize the content-rich web data.

2.1. The Hierarchical Model for Web Data

As shown in Fig. 1, the three-tier model consists of (1) physical pages, single physical files, (2) logical pages, sets of physical files linked or embedded together as a semantic unit, and (3) topics: grouped logical pages relevant to the same subject. Each abstraction level corresponds to a different use case.

Physical pages, also known as web objects, are single physical files that can be identified by their URLs. A physical page can either be a normal HTML file or an embedded media file used as a component of a web page. All web-caching policies developed so far deal only with physical pages and replacement decisions

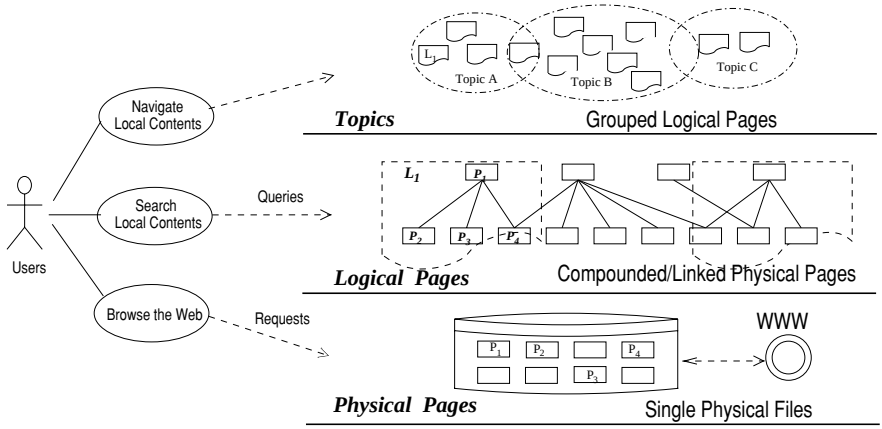

Fig. 1. Hierarchical model for content-aware cache management.

are primarily based on past usage and other physical properties, for example, frequency or recency of usage, size, and type of object files.

Logical pages are built upon one or more physical pages. We distinguish two kinds of logical pages: the *basic logical pages* and *coupled logical pages*. A basic logical paper is composed of a physical page together with its all embedded media components, if any. A coupled logical page consists of multiple directly linked and closely related basic logical pages: for example, a hypermedia article with one index page and several component basic logical pages, which should be viewed as a single logical unit. A logical page is a suitable information unit for searching and classifying. The idea of using linked pages as an information unit is from Li and Wu (1998), where a query can be evaluated even when queried terms are scattered across several linked pages. Since determining coupled logical page may be computationally complex, currently we only consider the basic logical page in the rest of the paper.

A topic is a higher-level abstraction of web data than logical pages. A set of logical pages relevant to a topic form a topic. Logical pages are categorized based upon various features such as keywords, home server, and language. Well-organized topics support navigation of the cache contents. For example, if a user is interested in 'usability', research, he/she can navigate the corresponding topic for information of interest. A topic may consist of a number of logical pages, while a logical page can also join into more than one topic depending on the categorization.

Similarly, a logical page consists of one or more physical pages. A physical page can belong to multiple logical pages. For example, a navigation panel may appear in a fixed position in many logical pages. A logical page will disappear with the last constituent physical page being deleted. A topic, however, will still exist even if it has no logical pages.

2.2. Implementation of the Hierarchical Model

The hierarchical model has built a conceptual framework for efficient management of web contents. To implement this model, we develop an architecture that enables

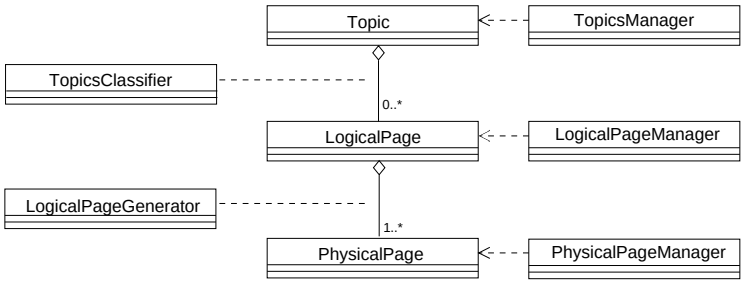

Fig. 2. Architecture for implementation of the hierarchical model.

active access to cache content and supports content-aware cache management. The architecture, composed of three managers corresponding to different abstraction levels, is illustrated in Fig. 2.

The physical page manager (PPM) acts as a simple cache manager, servicing the incoming requests and maintaining the cached physical documents in a priority queue. The logical page manager (LPM) is responsible for generating and maintaining the set of logical pages. Since links are important metadata for hypermedia documents, the 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 the LPM is to maintain logical pages, defining priority among different logical pages for cache replacement. Logical page replacements are based on the reference history as well as the relevance between the page and the topic cluster.

The topics manager (LPM) is at the highest abstraction level that classifies logical pages into given 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 replacement begins with choosing a topic that is least popular/important among all others.

In summary, the process of content management under this architecture can be described as follows. When a physical page is retrieved from the original server, the logical page generator is used to generate one or more logical pages, if there are any. A topics classifier is then used to determine which category 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 can be 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 to definition of priority orders or replacement policies for all abstraction levels.

3. Indexing and Clustering of Web Contents

A large proxy cache is a repository of web content that not only costs much time and network resources to download but also takes time and work for users to discover. However, owing to the content transparency tradition of the buffering paradigm, the content of cached web documents has not yet been well utilized. On the one hand, performance of caching can hardly be further improved owing

to the steady increase of storage capacity. On the other hand, a majority (above 60%) of web data is generally stored and replaced without any use, leading to a heavy waste of system information resources (Rizzo and Vicisano, 1998). In this section, we describe the content-aware support for users to actively make use of the most valuable resources that are locally available from the cache storage. We provide two facilities to this end: the search engine and the content directory, based on up-to-date indexes.

3.1. Indexing of Cache Content

The TF/IDF (term frequency/inverse document frequency) is the most widely used weighting scheme used for indexing documents in information retrieval systems. In this scheme the weight of each term is computed and then the documents are indexed based on the weights of these terms. Suppose t denotes the term (i.e., keyword and/or phrase), d denotes a document, N denotes the total number of documents, $TF_{t,d}$ (term frequency) denotes the occurrence of term t in document d, and DF_t (document frequency) denotes the number of documents that contains term t. The inverse document frequency IDF_t for term t and $w_{t,d}$, the weight of term t in document d, are defined as:

$$IDF_t = \log(N/DF_t) + 1 \tag{1}$$

$$w_{t,d} = TF_{t,d} \cdot IDF_t \tag{2}$$

As we can see, this indexing method uses syntactical information rather than semantics of documents. For example, if we rearrange order of words in a document, then the resultant document may be scored as high as the original one. This is inevitable unless more semantic information is used in evaluating a document.

Let keyword vector

$$q = ((t_1,\ w_1),\ (t_2,\ w_2), \cdots ,(t_k,\ w_k))$$

represent the information needs of a user, where, t_i and w_i $(i = 1, 2, \cdots, k)$ are keyword and importance of the keyword respectively. $w_i \in \{1, 2, 3\}$. Given q and document d, and $w_{t_i,d}$ is the TF/IDF weight for keyword t_i in document d as defined in formula 2, we use *cosine vector similarity* formula to compute the semantic distance (i.e., similarity) (Salton and Buckley, 1997).

$$\text{similarity}(d,\ q) = \frac{\sum_{i=1}^{k}(w_i \cdot w_{t_i,d})}{\sqrt{\sum_{i=1}^{k}(w_i)^2 \cdot \sum_{i=1}^{k}(w_{t_i,d})^2}} \tag{3}$$

We extend the basic scoring scheme by considering the popularity of a web document. The popularity of a document can be judged by determining the reliability (i.e., credibility) of the web document, how many other documents create a link to this document, and how often it is accessed. We refine the above formula as

$$\text{score}(d,\ q,\ RF_d) = \text{similarity}(d,\ q) \cdot e^{\alpha(RF_d - 1)} \tag{4}$$

where RF_d is the reference frequency of document d. α is a parameter and its value lies between 0 and 1, both inclusive. The user can set α to adjust the impact of the reference frequency. In the case $\alpha = 0$, it becomes the naive-form similarity

scoring. From (4), we can see that the weight of a document increases with RF_d. In the case that $RF_d = 1$, we get

$$\text{score}(d,\ q,\ RF_d) = \text{similarity}(d,\ q),$$

where score(d, q, RF_d) simply measures the relevance of document d to query q. However, when the document is proved to be popular and valuable ($RF_d > 1$), it becomes increasingly worthwhile to recommend it to users, so score(d, q, RF_d) $\geqslant$ similarity(d, q).

3.2. Keyword-Based Searching

Searching is a means to make users aware of relevant contents in a cache that are found and retrieved by other like-minded people. Searching as a new function improves the utilization of cached web content while facilitating information sharing among different users. Two kinds of searching are supported to facilitate retrieval of information satisfying different needs of users.

Persistent searching is provided for filtering web documents in which a user has persistent interest; for example, a researcher may have long-term interests in new documents on his/her research topic. Persistent searching is a continuous process to filter new documents for a specific user based on his/her user profile. A user profile is a vector of keywords (with weightings) representing the user's information needs or interests:

$$p = (\lambda,\ \delta,\ notification,\ url,\ email,\ \cdots, q)$$

where $q = ((t_1,\ w_1),\ (t_2,\ w_2), \cdots, (t_k,\ w_k)$ represents the user's information needs; λ is a threshold for deciding whether a new document is close enough to be selected. δ is the time interval in which the automatic searching will be performed. *notification* is used to specify whether and how to notify the user of the search results; it is a combination of one or more options from

$$\{none,\ normal,\ when\_updated,\ only\_fresh,\ only\_top\_N,\ by\_mail\}$$

where *url* is the location in which to keep the search results and all documents d with $score(d,\ q,\ RF_d) > \lambda$ will be returned. Both λ and δ are configurable parameters. Thus, when one wishes the search results to be updated on a daily basis, he/she can set $\delta = 24$ (hours).

Ad hoc searching is supported for casually searching the content of a cache. An agent is employed to mediate the search: if there are not enough results returned from the proxy cache, the agent will choose a suitable search engine to carry out the search, depending on the type of query the user issued.

3.3. Content Directory Based on Text Categorization

Content directory listing up-to-date resources of interest is a useful facility for people working on or studying similar subjects. The content directory also plays an important role in content-aware cache management because it corresponds to the clustered topics.

The system maintains a general category that includes everything not suitable in other categories. Privileged users can create new categories and train the

classifier with example pages. Each category has a priority for determining the order for cache replacement.

We implemented the classifier based on support vector machines (SVM), which have proved very well suited for learning text categorization (Joachims, 1998; Tong and Koller, 2000).

4. Cache Replacement Scheme Based on Hierarchical Content Management

According to the hierarchical model, a logical page can be selected for replacement only when it belongs to a candidate logical page, and this logical page is a member of a candidate topic, that is, a topic with least priority. In this section, we describe the content-aware extension to LRU-SP, a size-adjusted and popularity-aware LRU caching algorithm we have proposed in Cheng and Kambayashi (2000b).

4.1. LRU-SP: A Replacement Algorithm for Physical Pages

LRU(least recently used) is the most extensively used caching policy. The basic idea of LRU is that the more recently the data was used, the more possible it will be used again in the near future. Hence, LRU tries to keep recently accessed documents while replacing least recently used data to make space for new ones. LRU performs well in most cases in traditional paging scenarios; however, it is not suitable in the web context, since classical LRU cannot deal with varying object sizes; also it does not make use of reference frequency, a strong predictor of popularity. Thus, classical LRU should be extended to adapt the web context.

Size-adjusted and popularity-aware LRU (LRU-SP) is built upon one extension to LRU, namely, size-adjusted LRU (Aggarwal et al., 1999), which takes into account varying object sizes. In LRU-SP, we further incorporate reference frequency to differentiate the popularity of documents. The basic idea behind it 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 is

$$RF_i \cdot \frac{1}{\Delta T_{i,t} \cdot S_i}$$

where RF_i is the number of references to document i, $\Delta T_{i,t}$ is the time since the last reference to i, and S_i is the size of document i.

To choose a document with least benefit, we re-index all documents in cache in a non-decreasing order on values of $(S_i \cdot \Delta T_{it})/RF_i$, then greedily pick the highest index objects one by one and purge them from the cache until sufficient space is made:

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

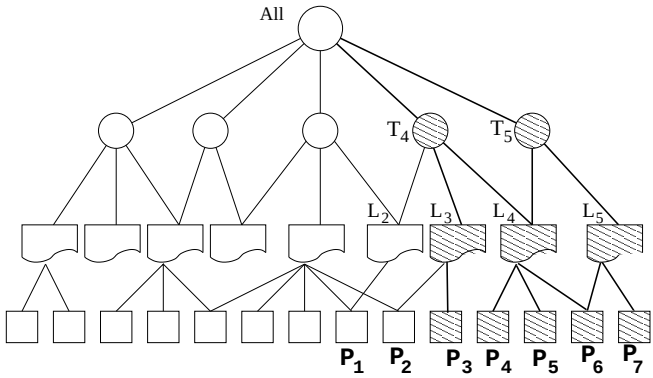

Fig. 3. Cache replacement based on the hierarchical model.

4.2. Cache Replacement Policies Based on the Hierarchical Model

Figure 3 depicts the replacement process. The first-round choice for a replacement begins from the topics. The least popular topics, for instance T_4 and T_5, will be chosen to continue the next-round choice. The priority of each topic is based on a predefined 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.

Once T_4 and T_5 are decided, where we can choose to replace some logical pages to make space, the next-round choice is a little complicated since choosing a logical page from the selected topics requires a trade-off between a page's relevance to the topic and the popularity known so far. Equation (4) is an example of this concern. Let the candidate logical page be, for example, $\{L_3, L_4, L_5\}$.

Now, the remaining decision is straightforward, since we just need a traditional caching policy applicable to most physical page contexts. As shown in Fig. 3, the scope for cache replacement is restricted, that is, $\{P_3, P_4, P_5, P_6, P_7\}$. Caching policies that can be used in this case include LRU (least recently used), LFU (least frequently used), size-adjusted LRU (Aggarwal et al., 1999) and any good algorithms developed so far.

4.3. LRU-SP+: A Content-Aware Replacement Algorithm for Proxy Caching

To verify the proposed approach, here we give a concrete algorithm that realizes the ideas developed so far. We employ LRU-SP as replacement algorithm for physical pages. The resultant algorithm, called LRU-SP+, makes replacement decisions from topic selection to logical page, physical page selections. Algorithm 1 describes the process.

The following content management functions are necessary to implement LRU-SP+:

- $p2L(p)$ returns a set of logical pages that the physical page p belongs to.
- $l2T(l)$ returns all topics that a logical page l belongs to.

- $t2L(t)$ returns all logical pages in topic t.
- $l2P(l)$ returns all physical pages in logical page l.

In addition, we need some functions implementing priority queues on different abstraction levels. $get\_topic\_least(T)$ is a function for selecting a topic with LEAST priority in topic set T. $get\_logical\_least(L)$ returns a logical page with LEAST priority in set L. $get\_physical\_least(P)$ returns a physical page with LEAST priority in set P. Let Ω be the set of all active topics in cache.

Algorithm 1 LRU-SP+ Content-Aware Replacement Algorithm

Require: $space$ = unused cache space
1: **for** each request q for some physical page, p_0 **do**
2: **if** p_0 is in cache **then**
3: return a copy of p_0
4: **else**
5: Retrieve p_0 from origin server;
6: $t_0 = get\_topic\_least(\Omega)$;
7: $L_0 = p2L(p_0)$; {get the logical pages p_0 belongs to}
8: $T_0 = \cup_{l_0 \in L_0} l2T(l_0)$; {Categorize logical pages to topics}
9: $max\_priority = \max\{priority(t) | \forall t \in T_0\}$
10: **if** $max\_priority \geqslant priority(t_0)$ **then**
11: $t =$ topic with $max\_priority$;
12: $L = t2L(t)$;
13: **while** $size(p_0) > space$ and $\neg empty(L)$ **do**
14: $l = get\_logical\_least(L); P = l2P(l)$;
15: **while** $size(p_0) > space$ and $\neg empty(P)$ **do**
16: $p = get\_physical\_least(P)$;
17: $remove\_physical(p, P)$;
18: $space+ = size(p)$
19: **end while**
20: **if** $empty(P)$ **then**
21: $remove\_logical(l, L)$;
22: **end if**
23: **end while**
24: LRU-SP(l_0, p_0); {cache p_0 within l_0}
25: **end if**
26: **end if**
27: **end for**

5. Performance Evaluation and Discussion

Content management enhances proxy caching with several important features. The first feature is content-based cache control. The hierarchical management of web data makes it feasible for administrators to control how cache resources are used according to the content. For example, companies often desire to use system resources in fast information retrieval related to their business, rather than material on leisure and recreation. It is difficult to realize this control in traditional caching schemes that only deal with physical pages. By our approach, it can simply be done by creating a topic for that content and setting it a lower priority.

Table 1. Profiles of trace datasets

Dataset	Requests	Bytes	Corpus	HR_{max}	BHR_{max}
KAMB	873,824	23.6 GB	21.3 GB	0.251	0.098
NLANR	1,848,319	21.0 GB	None	0.228	0.245

The second feature is active utilization of cache content. People often take it for granted that caching can only function in the background and the access patterns are independent of the cache's existence. As cache content can only be accessed passively and blindly without informing users what is available there, caching performance will stay at a low level no matter how large the cache size has been set and how 'smart' the replacement algorithm has been designed. However, content management techniques introduced in this paper offer an extra chance to improve caching performance by changing access patterns.

The third and most significant feature is enhanced replacement policies. Replacement policies such as LRU-SP+ can easily exploit knowledge of users' interests specified through topic and topic priority and can be more smart that replacement policies that are based only on analysis of past usage. In subsequent sections, we will experimentally evaluate this feature. In particular, we will compare LRU-SP+ with other usage-based algorithms.

5.1. Experimental Model

First, in order to differentiate value of topics, 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 measures 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

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

In addition, we also use *hit ratio* (HR) as in most caching schemes to evaluate performance:

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

We use the KAMB dataset, to be described in the next section, as input to drive the simulator. The simulator is based on the architecture discussed in Section 2, in which a new document (physical page) forms or is added to a logical page. The logical page is indexed and clustered to one or more topics. The cache management is based on LRU-SP+.

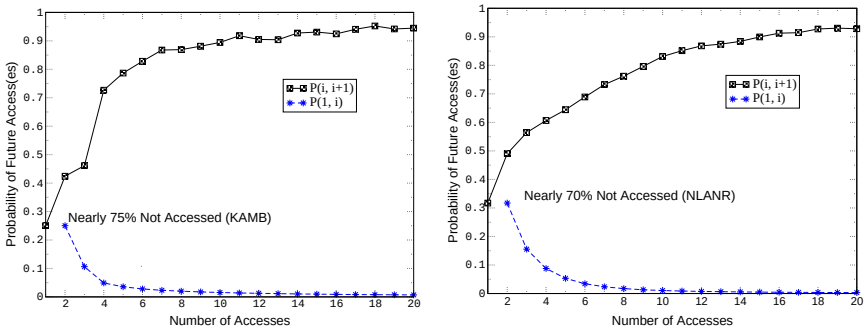

Fig. 4. Reuse ratios of cached documents as a function of number of references.

5.2. Data Collections

5.2.1. Basic Profiles of Experimental Data

The dataset NLANR in Table 1 is a one-week top-level caching proxy trace and is publicly available.[1] This dataset contains 1,848,319 requests with a total of 21.0 GB of web data, where unique data is 15.9 GB with a maximum hit ratio 0.228 and byte hit ratio 0.245. Since, we can not get the web data corresponding to NLANR access trace, we will unfortunately not perform topic analysis.

The KAMB dataset includes access logs as well as the corresponding web data collected from the Squid proxy server in our laboratory. The logs keep all of the 873,824 requests for a total of 23.6 GB of web data, where unique data is 21.3 GB in size. The maximum hit ratio is 0.251 and byte hit ratio is 0.098. The possible reason for such a low byte hit ratio could be that the laboratory proxy rarely sees repeated requests for large documents(larger documents may be saved to avoid time-consuming downloading), whereas the NLANR proxy sees requests from lower-level caching servers which may avoid caching larger documents, so requests for larger documents go up to the NLANR proxy server.

5.2.2. Popularity Differences between Topics

In Table 2, we list the major topics to be considered according to the status of the laboratory. The priority between topics is assigned based on significance and the population interested in that topic.

5.2.3. Around 60% of Documents are not Reused

Our analysis found that most of documents are accessed only once over trace period. Fig. 4 depicts how many documents will be reused in the future given they have been used several times. $P(i, j)$ is defined as allow:

$$P(i,j) = \frac{\|D_j\|}{\|D_i\|}, \quad (1 \leqslant i < j)$$

where D_i is the set of documents that has been accessed AT LEAST i times. $\|D_i\|$ is the size of the set. The $P(i, j)$ defines the *reuse ratio* for a specific

[1] ftp://ircache.nlanr.net/Traces

Table 2. Topics and their priority(topic significance)

Topics	Description	Priority	Weight
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

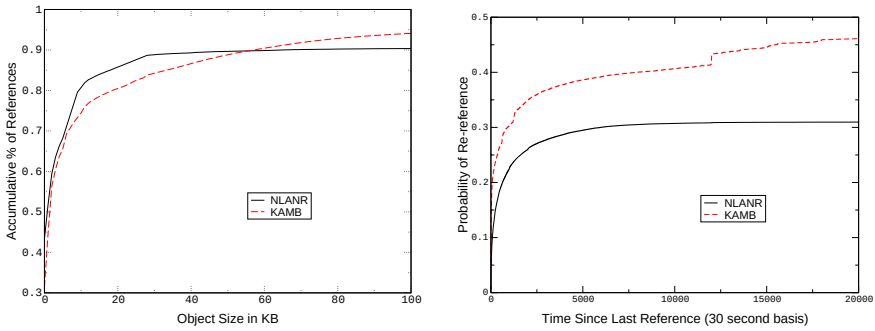

Fig. 5. Reference probability dependent on object sizes and recency of references.

set of documents. Note, D_1 actually represents the whole set of documents in consideration (all retrieved documents have been accessed at least once),). $P(1,2)$ is the reuse ratio for all documents in consideration. $1 - P(1,2)$ is the *no-use ratio*, that is, how many documents are not reused since its first access. In dataset NLANR, the *no-use ratio* is about 70% , while in KAMB it is near 75%! Even when considering the margin errors (those which have been accessed before trace beginning and those which will be accessed after trace finish), the *no-use ratio* can still be above 60%! Thus, it is significant progress if we can make the best use of this web data.

5.2.4. Access Frequency, Recency and Size are Useful Predictors

In addition, Fig. 4 also shows that the more frequently a document has been used, the more possible it is that it will be used again in the future. The plots of $P(i, i+1)$ go up quickly when the access time is less than 10, which means access time is a strong predictor to future usage. Figure 5 (left) shows the correlation between access probability and object sizes. It is noticeable that documents less than 20 KB in the NLANR dataset are more likely to be accessed than in the KAMB dataset. The distribution of sizes also shows some locality: objects below a small threshold, say 10 KB, comprise a majority of accesses. Figure 5 (right) shows the accumulative distribution of access probability versus recency of access, indicating that the more recently an object is accessed, the more likely it will be accessed more times in the near future.

5.3. Algorithms for Comparison

In our experiments, least relative value (LRV) (Rizzo and Vicisano, 1998) is used as a baseline algorithm for comparison. The designers of LRV developed an elaborate function handling various characteristics of web objects, such as object size, access frequency, and recency. The content-blind LRV performs better than most existing caching algorithms. An efficient content management scheme is given to LRV, which 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 remaining groups are FIFO lists. Final decisions are based on LRV function. LRU-SP+ is compared with LRV, as well as LRU-SP, in terms of hit ratios and profit ratios.

5.4. Results and Analysis

The experimental results under different datasets are shown in Fig. 6. The sub-figures 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 the cache, and profit ratio, a weighted version of hit ratio by considering the significance factor of topics, are significantly better than the baseline algorithm LRV. LRU-SP+ also improves LRU-SP remarkably.

5.4.1. Topic-Based Priority Results in Higher Hit Ratios

First, LRU-SP+ performs about 20% better than LRV and nearly 10% better than LRU-SP in terms of hit ratio, as shown in Fig. 6 (left). This is because in our 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 content. Documents irrelevant to these topics are rarely accessed, and are less likely cached due to the lower priority; even cached, they may quickly be replaced. The topic-based priority guarantees popular content being cached long enough, whereas less popular ones will not occupy cache space.

5.4.2. Significance Factor Guarantees Valuable Documents are Cached Prior to Others

In terms of profit ratios, LRU-SP+ outperforms LRV by a factor of 30% or so, as shown in Fig. 6 (right). This is reasonable because in our significance factor assignment content with higher priority is assigned a relatively high profit. This factor further enlarges the benefits obtained from the hit ratio. LRU-SP+ also improves LRU-SP in a similar way. The dash-dotted plot in Fig. 6 (right) represents profit rates of LRU-SP. LRU-SP+ performs nearly 20% better than LRU-SP in profit ratios.

6. Related Work

While a long list of web caching schemes have bee proposed in recent years, only a few of them have discussed the explicit utilization of semantic information for management of web documents. Bestavros et al. (1996) suggested using more

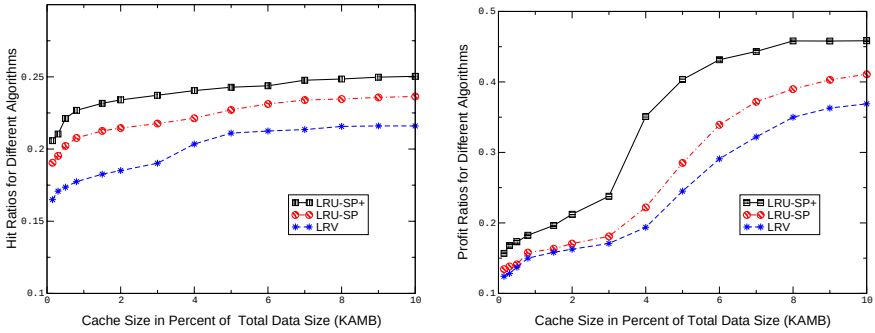

Fig. 6. Experimental results.

application-level information including document content in cache management, but they do not provide any ideas how to realize this. Rizzo and Vicisano (1998) discussed several parameters that should influence the probability of re-access, including the document's source, client requesting the document, file type, etc. They presented *LRV* (least relative value) based on an elaborate benefit/cost function of access recency, frequency, and document size. A difficult task for LRV is to estimate the probability of access to documents with only one access, which represents nearly 60% of residents in cache (Rizzo and Vicisano, 1998). They adopt a document size-based scheme to decide the probability of re-access when an object has been accessed for the first time. Shim et al. (1999) propose *LNC-R-W3-U*, a unified algorithm that combines cache replacement policy and cache consistency policy. The replacement policy in LNC-R-W3-U is based on *LRU-K* (Neil et al., 1993), an extension to LRU (least recently used) caching policy where the cache decision is based on the last k accesses, in contrast to LRU, which considers only the last *one* access.

Studies on large-scale caching as a solution to scale content delivery are an active field of research. The LSAM proxy cache (Touch and Hughes, 1998) is user-based content-aware 'virtual cache', using multicast push of related web pages, i.e. automatically selected interest groups, to load caches at natural network aggregation points. INTELSAT Internet Delivery System (IDS) (Chen et al., 1999) is a content delivery system based on a warehouse–kiosk model, which provides global access and Internet wormholes via a fleet of INTELSAT satellites. Web content such as cacheable HTTP, FTP, and streaming objects are fetched or pushed both actively and reactively into a central repository cache via intelligent Web agents. Fresh objects are constantly sent via IP multicast reliably to registered Kiosk caches. Distributed web caches in the kiosks offer content to their local users directly with improved quality of service and less bandwidth cost.

7. Conclusion and Future Work

Proxy caches are extensively deployed as a major way to improve web performance, and large proxy caches are increasingly important for efficient utilization of web content and network resources. In this paper, we have proposed a framework of a proxy caching where both users and cache manager are aware of the content of web documents. From the cache users' view of point, a large proxy cache is a repository of web documents shared by a set of users. From the cache

point of view, the content of web documents is collected by users with special information needs; thus by extracting popular topics of this content, a cache can predict the interests and needs of users and then make replacement decisions based on this kind of knowledge. The main contributions of this paper are as follows:

1. To the best of our knowledge, we have proposed for the first time using a proxy cache as a shared information repository, rather than simply a collection of physical data. A hierarchical data model was developed to exploit cache content and its semantic information.
2. Based on this data organization, we have provided facilities for easily finding useful information in the cache so as to maximize information sharing.
3. We also designed and implemented LRU-SP+, a content-aware algorithm for web proxy caching.

Most importantly, in this paper, we are finding a new approach to incorporating content management with performance tuning techniques. We believe that this is a promising way to solve the problems caused by the exponential growth of web size and Internet traffic. Also, some more experiments are required to verify the scheme for computing content-based popularity, and it is also necessary to develop facilities for using cache as a web warehouse to enable more active use of cache content.

Acknowledgements. The authors thank the anonymous reviewers for their detailed comments and valuable suggestions. We would also like to thank Mukesh Mohania for his helpful advice and productive discussion during his visit to our laboratory.

References

Aggarwal C, Wolf J, Yu P (1999) Caching on the World Wide Web. IEEE Transactions on Knowledge and Data Engineering 11(1): 95–107

Barish G, Obraczka K (2000) World Wide Web caching: trends and techniques. IEEE Communications Magazine, Internet Technology Series. http://www.isi.edu/people/katia/cache-survey.pdf

Bestavros A, Carter R, Crovella M, Cunha C, Heddaya A, Mirdad S (1996) Application level document caching in the Internet. In IEEE SDNE'96: Second international workshop on services in distributed and networked environments. Whistler, British Columbia

Cao P, Irani S (1997) Cost-aware WWW proxy caching algorithms. In Proceedings of the 1997 USENIX symposium on Internet technology and systems, pp 193–206. http://www.cs.wisc.edu /cao/publications.html

Chen H, Abrams M, Johnson T, Mathur A, Anwar I, Stevenson J (1999) Wormhole caching with HTTP PUSH method for a satellite-based web content multicast and replication system. In Proceedings of 4th international WWW caching workshop, San Diego, CA. http://www.ircache.net/Cache/Workshop99 /Papers/chen-html/

Cheng K, Kambayashi Y (2000a) Advanced replacement policies for WWW caching. In Proceedings of the first international conference on Web age information management (WAIM'2000), LNCS 1846. Springer, Berlin, pp 239–244

Cheng K, Kambayashi Y (2000b) 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). IEEE Computer Society Press, Los Alamitos, CA, pp 48–53

Cheng K, Kambayashi Y (2000c) Multicache-based content management for Web caching. In Proceedings of the first international Web information systems engineering (WISE'00). IEEE Computer Society Press, Los Alamitos, CA, pp 42–49

Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In European conference on machine learning, ECML98. TR 23, University of Dortmund, Lehrstuhl Informatik III,97

Lawrence S, Giles C (1999) Accessibility of information on the web. Nature 400: 107–109

Li W, Wu Y (1998) Query relaxation by structure for document retrieval on the web. In Proceedings of 1998 advanced database symposium, Japan, pp 19–26
Nielsen J (1998) Nielsen's law of Internet bandwidth. The Alertbox, http://www.useit.com /alertbox/980405.html
Neil E, Neil P, Weikum G (1993) The LRU-K page replacement algorithm for database disk buffering. In Proceedings of ACM SIGMOD international conference on management of data, New York pp 297–306
Rizzo L, Vicisano L (1998) Replacement policies for a proxy cache. Technical report rn/98/13, Department of Computer Science, University College London. http://www.iet.unipi.it/ luigi/caching.ps.gz
Salton G, Buckley C (1997) Term-weighting approaches in automatic text retrieval. In Jones K, Willett P (eds) Readings in information retrieval. Morgan Kaufmann, San Mateo, CA, pp 323–328
Shim J, Scheuermann P, Vingralek R (1999) Proxy cache design: algorithms, implementation and performance. IEEE Transactions on Knowledge and Data Engineering 11(4): 549–562
Tong S, Koller D (2000) Support vector machine active learning with applications to text classification. In Langley P (ed) Proceedings of ICML-00, 17th international conference on machine learning, Stanford, CA. Morgan Kaufmann, San Mateo, CA
Touch J, Hughes A (1998) The LSAM proxy cache: a multicast distributed virtual cache. In Proceedings of the third international WWW caching workshop, Manchester, UK. Also in Computer Networks and ISDN System 30(22–23), 25 November 1998, pp 2245–2252
Williams S, Abrams M, Standridge C, Abdulla G, Fox E (1998) Removal policies in network caches for World-Wide Web documents. In Proceedings of ACM SIGCOMM'97, Stanford, CA, pp 293–305
Wooster R, Abrams M (1997) Proxy caching that estimates page load delays. In Proceedings of the sixth international World Wide Web conference, pp 325–334. Also in Computer Networks and ISDN Systems 29: 1497–1505

Author Biographies

Kai Cheng received a B.S. in Computer Science from Nanjing University, Nanjing, China, in 1988, and an M.S. in Computer Science from Wuhan University of Hydraulic and Electrical Engineering (WUHEE), Wuhan, China, in 1991. From 1991 to 1998, he was a lecturer in the Department of Computer Science at WUHEE. He is currently a Ph.D. student in the Department of Social informatics, Graduate School of Informatics, Kyoto University, in support of a Japan Government Scholarship. His research interests include web databases, networked content delivery, fuzzy information processing and machine intelligence.

Yahiko Kambayashi received B.S., M.S., and Ph.D. degrees in Electronic Engineering from Kyoto University, Kyoto, Japan, in 1965, 1967, and 1970, respectively. During 1971–1973 he was a Visiting Research Associate at the University of Illinois, Urbana. From 1984 he was a Professor at the Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka, Japan. Since 1990, he has been a professor at Kyoto University. Currently he is a full professor at the Department of Social Informatics, Graduate School of Informatics. He has been working in databases and related areas since 1976. He has been a member of VLDB endowment, and chair of SIGDBS (database) of IPSJ (Information Processing Society of Japan). His current interests include virtual organization in a network and performance improvement of web systems. He is a Fellow of the IEEE and of the IPSJ and a member of ACM.

Correspondence and offprint requests to: Kai Cheng, Department of Social Informatics, Graduate School of Informatics, Kyoto University, Yoshida Sakyo, Kyoto 606-8501, Japan. Email: chengk@db.soc.i.kyoto-u.ac.jp