Using Database Technology to Improve Performance of Web Proxy Servers

Kai Cheng, Yahiko Kambayashi
Department of Social Informatics
Graduate School of Informatics, Kyoto University
Sakyo Kyoto 606-8501, Japan
{chengk, yahiko}@kuis.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 the cache at a proxy server as a web warehouse with data organized in a hierarchical model, similar to data organization in database systems. The hierarchical model consists of physical pages, logical pages and topics, corresponding to different abstraction level. Based on this model, we then develop searching and topic navigation facilities for easily finding relevant contents in cache. By defining priorities on each abstraction level, 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 most relevant, popular and high quality contents. 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.

Keywords

Database, proxy server, web caching, performance, text categorization

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 [9]. 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.

Although there is plenty of work done in last few years to improve caching schemes, only few of them have considered the utilization of semantic information explicitly for management of web data. A. Bestavros et al [3] suggested using more application-level information including document content in cache management, but they failed to realize this. [11] discussed several parameters that should influence the probability of re-accesses, including document's source, client requesting the document, file type etc. The authors 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 accessing documents that have been accessed only once, which represent nearly 60% of residents in cache. They adopt a document size based scheme to decide the probability of re-access when an object have been accessed for the first time. [13] proposed a unified algorithm *LNC-R-W3-U*, which combines cache replacement policy and cache consistency policy. LNC-R-W3-U is based on *LRU-K* [10], an extension to LRU (Least Recently Used) caching policy where the cache decision is based on last K accesses, in contrast with LRU, which cares only last *one* access.

We have noticed the importance of exploiting semantic information and user preferences for cache management. In [4], 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 [6], 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. We then develop searching and topic navigation facilities for users to quickly find what they usually would not be aware so as to keep the users informed of fresh information in cache. Furthermore, by defining priorities over topics, logical pages and physical pages respectively, the cache manager can make replacement decisions hierarchically based on the topics, logical

WebDB '2001, 2001 ACM SIGMOD Workshop on the Web and Databases, Santa Barbara, CA

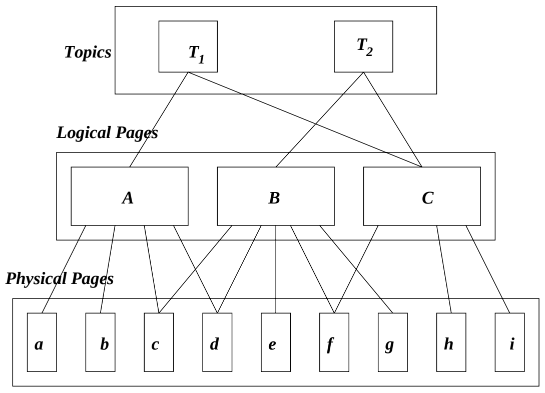

Figure 1: Hierarchical model for web data

pages and physical pages. As a result, the cache can always keep the 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. In section 3, we develop methods to enable searching and navigating the cache space. Section 4 presents a content-aware replacement algorithm, LRU-SP+. Section 5 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* is called to generate one or more logical pages if there are any. Then a *classifier* is 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 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 Hierarchical Architecture for Implementation

The data abstraction and management lead to a hierarchical system architecture as shown in Figure 2. 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 abstract 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 begin with choosing a topic that is least popular /important among all others.

3. ACTIVE ACCESS TO CACHE CONTENTS

Large proxy cache is a repository of web contents that not only costs much time and network resource to download them but also takes time and work for users to discover them. However, due to the content-transparency tradition of buffering paradigm, the content of cached web documents has not yet been well utilized. On the one hand, performance of caching can hardly be significantly improved due to the steady increase of storage capacity. On the other hand, a majority, above 60%, of web data are generally stored and replaced without any use, leading to a heavy waste of system-information resources. In this section, we describe the content-aware support for users to actively make use of the most valuable resources that

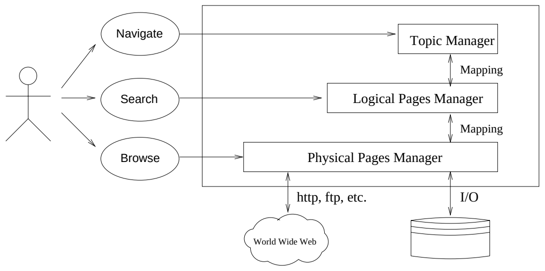

Figure 2: The hierarchical architecture for proxy caches that support bi-directional content-awareness: cache knows whether a document belongs to popular/important topics, users know what are available locally in cache

are locally available from the cache storage. We provide two facilities to this end: search engine and 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 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:

$$w_{t,d} = TF_{t,d} \cdot IDF_t, \quad IDF_t = log(N/DF_t) + 1 \tag{1}$$

As we can see this indexing method uses syntactical information rather than semantics of documents. For example, if a document is not in a proper order or it is a meaningless but it has all the terms, this document may be scored high. This can be avoided if the information about the credibility of documents is also included while computing the weights of terms.

Let keyword vector

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

represents 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 1, we use *cosine vector similarity* formula to compute the semantic distance (i.e. similarity) [12].

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

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 web document, how many other documents create link to this document, and how often it is accessed. We refine the above formula as:

$$score(d,\ q,\ RF_d) = similarity(d,\ q) \cdot e^{\alpha(RF_d - 1)} \tag{3}$$

where RF_d is reference frequency of document d. α is a parameter and its value lies between 0 and 1, both inclusive. User can set α to adjust the impact of reference frequency. In case $\alpha = 0$, it becomes the naive form similarity scoring. From (3), we can see, the weight of a document increases with RF_d. In case that $RF_d = 1$, we get

$$socre(d,\ q,\ RF_d) = similarity(d,\ q),$$

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

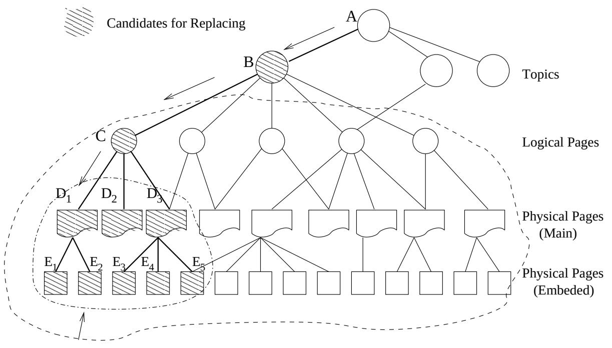

Figure 3: Cache replacements from topics down towards physical pages

3.2 Keyword-Based Searching

Searching is a means to make users aware of relevant contents in cache that are found and retrieved by other like-minded people. Searching as a new function improves the utilization of cached web contents 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 that a user has persistent interest, for example, a researcher may have long-term interests in new documents on her/his research topic. Persistent searching is a continuous process to filter new documents for a specific user based on her/his 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 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 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

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

url is the location to keep the search results and All document d with $score(d,\ q,\ RF_d) > \lambda$ will be returned. Both λ and δ are configurable parameters. For, when one wish the search result be updated on a daily basis, s/he can set $\delta = 24$ (hours).

Ad hoc searching is supported for casually searching the content of cache. An agent is employed to mediate the search: if there are no enough results returned from proxy cache, the agent will choose a suitable search engine to carry out the search, depending on the type of query user issued.

4. CONTENT-AWARE CACHING

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.

4.1 Multi-layered Cache Replacement Policies

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 3, 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 [5], which has proved well suited for caching physical pages. In the following, we will review the LRU-SP algorithm.

4.2 LRU-SP: A Replacement Policy for Physical Pages

LRU-SP (Size-adjusted and Popularity-aware LRU) is an extension to LRU (Least Recently Used) proposed by K. Cheng et al in [5]. 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.

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

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

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+*. Let Ω be the set of all active topics in cache. In Algorithm 4.3, we use the following predefined functions:. $get\_topic\_least(T)$ chooses 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. $p2L(p)$ returns all logical pages that a logical page p would belong to. $l2P(l)$ returns all physical pages belonging to a logical page l. $t2L(t)$ returns all logical pages contained in topic t.

Algorithm 1 LRU-SP+ Proxy caching replacement algorithm

```
Require: $space$ = unused cache space
 1: for each request $q$ for some physical page, $p_0$ do
 2:   $t_0 = get\_topic\_least(\Omega)$;
 3:   if $p_0$ is in cache then
 4:     return a copy of $p$
 5:   else
 6:     Retrieve $p_0$ from origin server
 7:     $L = p2L(p_0)$; { Find logical pages of $p_0$}
 8:     $T = \cup_{l \in L} l2T(l)$; { Get all topics that logical pages $L$
        should belong to}
 9:     $max\_priority = \max\{priority(t) | \forall t \in T\}$
10:     if $max\_priority \ge priority(t_0)$ 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
```

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[7, 8]. 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.

5. EXPERIMENTATIONS

5.1 Performance of content-aware caching

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.

1. Cost saving for resource discovery. 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.
2. High level performance tuning and control for system administrators. One can now play around tangible information units such as topics, logical page
3. Cost saving in time and network bandwidth. This is the basic feature of most caching schemes.

It is important to incorporate the quality and relevance of information with the conventional benefit/cost model. For example, business companies may may quick response times for the information relevant to their business, rather than material about leisure and recreation. In content-aware schemes, we do not treat all hits identically, instead we put emphasis on priority of topics.

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 [11]. 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.

5.2 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.

5.2.1 *Performance measurement*

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

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

In addition, we also use *hit ratio(HR)* as in most caching schemes to evaluate the 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 previously discussed architecture, in which 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+.

5.2.2 *Data collections*

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 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)

5.3 Results and analysis

The experimental results under different datasets are shown in Figure 4. 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.

5.3.1 *Topic-based priority results in high hit ratios*

First, LRU-SP+ performs about 20% better than LRV in terms of hit ratio as shown in Figure 4(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.

5.3.2 *Significance factor benefits more*

In terms of PRs, LRU-SP+ outperforms over LRV by a factor of 30% or so as shown in Figure 4(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.

6. CONCLUSIONS AND FUTURE WORK

Large proxy caches are increasingly important for efficient utilization of web contents and network resources. In this paper, we have

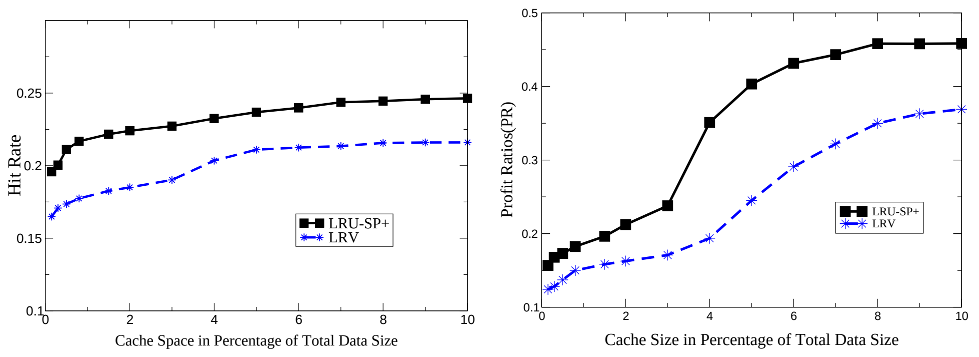

Figure 4: Experimental results

proposed a framework of a proxy caching where both users and cache manager aware about the content of web documents. From the cache users view of point, large proxy cache is a repository of web documents shared by a set of users. From the cache view of point, the content of web documents are collected by users with special information needs, thus by extracting the 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:

1. To the best of our knowledge, we have proposed for the first time to use proxy cache 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.
2. Based on this data organization, we have provided facilities for easily finding useful information in the cache so as to maximize the 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 the 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 contents.

7. 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] A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad. Application Level Document Caching in the Internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 297–306, New York, 1993.

[11] 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.

[12] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. In K. S. Jones and P. Willett, editors, *Readins in Information Retrieval*, pages 323–328. Morgan Kaufmann, 1997.

[13] J. Shim, P. Scheuermann, and R. Vingralek. Proxy Cache Design: Algorithms, Implementation and Performance. *IEEE Transactions on Knowledge and Data Engineering*, 1999.