Signature Cache: A Light Weight Web Cache Indexing Structure

Yuping Yang and Mukesh Singhal
Department of Computer and Information Science
The Ohio State University
2015 Neil Avenue,
Columbus, OH 43210-1277
{yangy,singhal}@cis.ohio-state.edu
(614) 292-5839

July 31, 1998

Abstract

Current trend in Web cache research is to have Web caches sharing their contents to improve the hit ratio. High performance Web cache sharing requires use of access indexes for Web caches to reference each other. The challenges the design of access indexes for sharing Web caches face are the huge size, dynamic nature of Web cache contents, and of course, high access speed.

Recently proposed summary cache scheme [9] scheme uses relatively small indexes for sharing Web caches to reference each other. We improved the summary cache scheme and propose a signature cache scheme. Instead of "repairing" existing access indexes, signature cache scheme builds new indexes to accommodate changes of Web cache contents. This scheme simplifies the maintenance of the indexes and significantly reduces the size of counters. The size of the index is further reduced by a semi-distributed index sharing mode. These improvements result in orders of magnitude reduction in the index size as compared to the summary cache scheme.

These benefits of the signature cache scheme come at a cost of slightly increased (one additional message in the worst case) response time as compared to that of the summary cache scheme. In applications where small size and ease of maintenance are favored over the response time, the signature cache scheme is a more attractive alternative than the summary cache scheme.

Key words: Web cache, signature, distributed, index, sharing.

1 Introduction

Web accesses often cross countries or even continents. A way to reduce network traffic and shorten the perceived response time of Web accesses is to cache frequently used Web pages in Web caches that are physically close to users. Large Web caches within Web proxies have been effectively used to reduce the network traffic [13]. The hit ratio of Web caches is critical to the Web access response time seen by end users and sharing Web cache contents is a way to boost the overall hit ratio for a group of Web caches.

Sharing of Web caches can be done in a straightforward way, e.g., when a Web page being looked for by a query does not exist in a Web cache, the query processor can search other Web caches in the group until the requested Web page is found in one of the Web caches in the group, or finally the query processor concludes that the Web page is not in this group of Web caches and a message is sent to the original server of this Web page for a fresh copy. This type of Web cache sharing was proposed in Harvest project [10, 11] and in the Internet Cache Protocol (ICP) [12]. Many institutions in many countries have created proxy caches using ICP to reduce network traffic over the Internet [9].

However, ICP protocol causes heavy network traffic among participating Web caches because for a cache miss, messages are sent to all other Web caches to find out whether the requested Web page is present in any of other Web caches. Use of access indexes can shorten the response time in accessing the Web caches and reduce the number of messages sent among Web caches. One of the recent proposals of indexes designed for Web cache sharing is summary cache scheme [9]. We make three important improvements over the summary cache scheme, namely, the semi-distributed index sharing mode, simplified index update operation, and simplified design of counters. We call the improved Web cache index design the "*signature cache*" scheme.

A summary cache [9] uses 0/1 bit-strings (Bloom filters, for short we call them filters) to code summary information of Web pages. Each URL is hashed to a fixed length filter and different URLs have different combinations of 0's and 1's in their filters. The content of each Web cache is partitioned. All filters in a particular partition of a particular Web cache are superimposed (bitwise-ORed) to a filter, partition filter, which represents (summarizes) all URLs in the partition. A query looking for a Web page (keyed by a URL) hashes the URL into a filter, query filter, and uses it to search all partition filters. If every "1" bit in the query filter is matched by a "1" bit in the same bit position of a partition filter, then it is possible (the possibility can be very high if the filter length is long enough) that the queried Web page is present in the partition. Otherwise, the queried Web page is not present in the partition. So, use of Bloom filters can speed up the search process because the size of Bloom filters is much smaller than the size of corresponding URLs.

All partition filters extracted from a Web cache form a filter file. The filter file of a

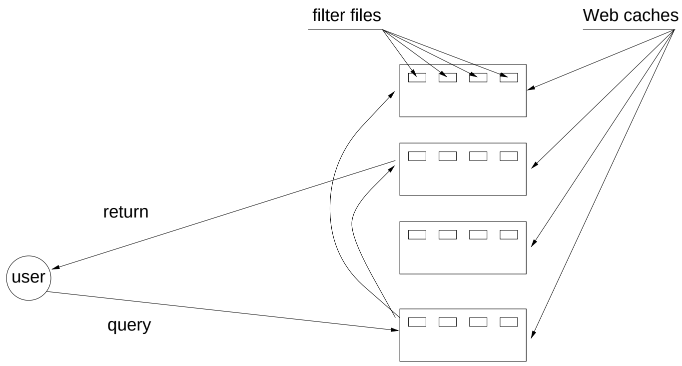

Figure 1: The fully distributed index sharing mode.

particular Web cache is sent to all other Web caches. If there are m Web caches, then each Web cache has $m-1$ filter files for $m-1$ other Web caches. A query directed to a particular Web cache first searches all filter files in that Web cache. This search produces a list of Web caches (candidate list) that possibly contain the desired Web page. If this Web page is not found locally, then the local Web cache is searched and when the Web page is found, it is returned to the end user who initiated the query. If this Web page can not be found locally and there are other Web caches on the candidate list, then messages are sent to those Web caches on the candidate list to search them. This is sketched in Fig.1.

For convenience of reference, we divide the messages over the network into two categories. A message that contains request for searching a Web cache is called a signal message and a message that contains found Web pages is called a data message. Obviously, a data message is usually much larger than a signal message.

The main contribution of the summary cache scheme [9] is that it reduces the number of signal messages by using Bloom filters to identify Web caches that are very likely (typically the chance is over 99.9% [9]) to contain the desired Web pages and send signal messages only to these identified Web pages.

The summary cache scheme copes with the dynamic nature of the Web contents by using counters for each bit in each partition filter. The following discussion is only for one partition filter and it's associated counters. A partition filter represents a group of URLs. Each bit of the partition filter may be set to 1 many times by different filters of different URLs. When a new Web page is added to a partition, a new filter (not the partition filter) is hashed from the URL of the Web page and is bitwise-ORed with the existing partition filter to obtain a new partition filter. The counters (recall that a counter is associated with each bit in the partition filter) corresponding to the positions of 1's of the newly hashed filter is incremented by 1. When an existing Web page is deleted from a partition, the counters corresponding to the positions of 1's of the filter of the deleted Web page is decremented by 1. When a counter is incremented from 0 to 1, the corresponding bit in the partition filter is set to 1 and when a counter is decremented from 1 to 0, the corresponding bit in the partition filter is set to 0. By using counters, the updates of Web pages in the partition can be precisely reflected into the partition filter.

This paper is organized as follows. Section 2 describes the the proposed signature cache scheme. Section 3 analyzes the index size reduction of the signature cache scheme over the summary cache scheme. Section 4 analyzes the filtering effect of using signature files through both probability analysis and experiments, and also analyzes the search response time of the signature cache scheme. Section 5 discusses the implication of the signature cache scheme to ICP. Finally, Section 6 concludes the paper.

2 Signature Cache Scheme

We improve over summary cache scheme in several aspects. First, we noted that the approach to put every Web cache's filter file into every other Web cache has a large space overhead of the index structure. We propose a semi-distributed index sharing mode to rectify this problem. Second, we noted that there is no inherent reason new Web pages being added to a Web cache have to go into any particular partition. The content of a Web cache is not partitioned in any meaningful way. The only requirement for partitioning is that the size of each partition has to be reasonable for hashing Bloom filters. So, we realized that the precise tracking of the change of Bloom filters is not really necessary. Based on this understanding we propose a simplified update operation for the Bloom filters. Third, for the simplified update operation, the one counter per bit position for a partition filter in the summary cache scheme is not needed and we propose a simplified counter design. The improved Web cache design with these improvements is called the signature cache scheme.

2.1 Semi-Distributed Sharing Mode

The fully distributed index sharing mode in the summary cache scheme uses large storage spaces, especially when the number of Web caches is large. For example, suppose there are 4,000 Web caches, each Web cache has 8 GB of Web page contents, and each Web page is 8KB. Then, each Web cache has 1M Web pages. Suppose every 50 URLs are grouped together to extract a partition filter and the length of the filters is 800 bits (the load factor is $800/50 = 16$), then the size of the filter file of a Web cache is about 2MB. Because every Web cache contains the filter files of all other Web caches, it takes $4,000 * 2M = 8GB$ disk space for storing all partition filters in each Web cache. In other words, the space to store filters is equal to the space to store all data in each Web cache.

To reduce index size, we propose a semi-distributed index sharing mode, i.e., only one Web cache, the coordinating Web cache, is elected to store all filter files from all Web caches. When a Web cache miss happens and this Web cache is not the coordinating Web cache,

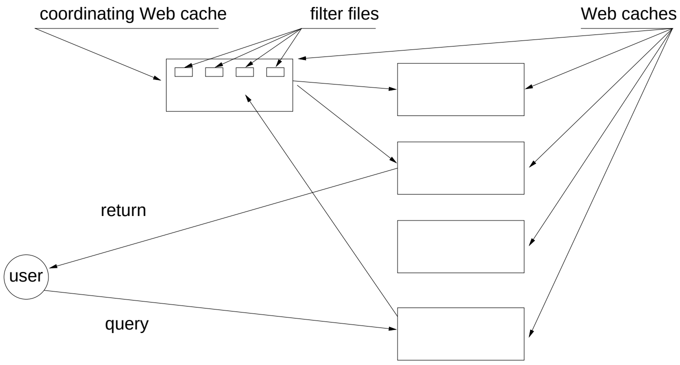

Figure 2: The semi-distributed index sharing mode.

a signal message is sent from the Web cache to the coordinating Web cache to search filter files and based on this search, a list of candidate Web caches will be searched to look for the desired Web page requested by the query. This is sketched in Fig.2. Because there is only one copy of each partition filter in the system, for the above scenario, the total disk space needed for storing all filter files in the semi-distributed sharing mode is only $4,000 * 2MB = 8GB$. This is much smaller when compared to $4,000 * 8GB = 32,000GB$ in the fully distributed sharing mode in the summary cache scheme.

2.2 Simplified Update Operations and Input Partition

There is no inherent reason why newly added Web pages to a Web cache have to go into some existing partitions. By directing all new Web pages into one partition, we avoid the work to update all existing partition filters and the counters for every bit of every filter in the filter file.

For a particular filter file (recall that all partition filters extracted from a Web cache form a filter file), our signature cache scheme has only one "input" partition and the rest of the partitions of the filter file are "stable" partitions. Once made, the partition filter of a stable partition never changes. A stable partition only allows deletion of Web pages from the partition and does not allow addition of a new Web page to it. All new Web pages are added to the input partition. Once the number of Web pages in the input partition reaches the average size (which can be pre-determined) of a stable partition, the input partition becomes one of the stable partition and a new input partition is created. With this approach, even if there is a large number of new Web pages being constantly added to the Web cache, there is only one partition, the input partition, and one partition filter, the partition filter for the input partition, are being modified. This approach drastically simplifies the maintenance work for the filter file.

2.3 Simplified Design of Counters

By using an input partition, counters are not needed for recording the addition of new Web pages to a Web cache. Counters are still needed for recording the deletion of existing Web pages from a Web cache. Since we are not going to modify the partition filter, counters are only used to signal whether "enough" Web pages have been deleted from a partition so that the partition filter should be re-make. One counter per partition filter is sufficient for our purpose.

In a signature cache, the counter for a partition is initialized to zero when the partition becomes a stable partition and the partition filter is made. The counter is incremented by 1 each time an existing Web page is deleted from the partition. When the counter for the partition filter is large enough, we know that enough number of Web pages have been deleted from the partition; so, we discard the partition filter and put all remaining Web pages into the input partition. This is equivalent to treat these Web pages as new Web pages to the Web cache. This approach greatly reduces the total size of counters. If the length of partition filter is 800 bits, then the total size of counters is only 0.125% (1/800) of that in the summary cache scheme.

2.4 Main Differences between Signature and Summary Caches

Note that Bloom filter is just another term for superimposed signature. So, in the rest of this paper we refer to signature and filter interchangeably. We choose to use the term signature to name our design of the Web cache because signature is a more popularly used term since 1980s [2, 3, 4, 5, 6, 7, 8]. The main differences between the signature cache scheme and the summary cache scheme are the different index sharing mode (semi vs. fully distributed), different updating schemes (updating only the input partition filter vs. updating all partition filters), and different counter designs (one counter per partition filter vs. one counter per bit per partition filter). These improvements result in a much smaller Web cache index which requires much less and simpler maintenance work.

3 Benefits of the Signature Cache

The main advantages of the signature cache scheme over the summary cache scheme are the small index size and ease of maintenance for index structures. The ICP protocol does not need any index structure. However, to share contents of a large number of Web caches, the network traffic overhead generated by using ICP protocol is unbearable. Summary cache scheme [9] drastically reduces the network traffic over the ICP protocol with the cost of maintaining an index structure in each participating Web cache. These index structure can be very large if a large number of Web caches are to share their contents. We made several

size of each Web cache	8GB
size of each Web page	8K
loader factor (number of bits per Web page)	16
size of filters contributed by each Web	2MB
size of each counter	4 bits
length of each filter	800 bits

Table 1: Parameters used in Fig.3, 4, and 5

improvements to reduce index size and simplify index update operations.

It is not easy to quantify the benefit of simplified index maintenance due to one counter per partition design and the approach to only update the input partition in a filter file. Apparently, the ease of maintenance of index structure is an important issue in practical Web cache implementations. Because all except one partition are stable partitions, the software to update the index structure when the content of Web cache changes is much simpler and the update operations cause much less random disk accesses.

Two factors contribute to the small size (as compared to the summary cache scheme) of the signature caches: semi-distributed signature index sharing mode and one counter per partition design. From Fig.3, we can see that the size reduction effect of using the semi-distributed index sharing mode as compared to the fully distributed sharing is very large. Fig.3 shows the change of filter size (not including the size of counters) with respect to the change of number of Web caches. The vertical axis in the figure is the total size of all filters (signatures) for all Web caches. The semi-distributed sharing mode has a drastically low space overhead because only one copy of each filter is kept in the system. Naturally, we are concerned about the cost of using the semi-distribute index sharing mode, i.e., the additional network traffic caused by sending an extra signal message to the coordinating Web cache. It turns out that the network traffic overhead of the extra signal message is quite low as discussed in Section 4.

The size reduction effect due to the one counter per partition design as compared to the one counter per bit design is plotted in Fig.4. The size of all counters in the one counter per bit design is not affected by the length of the filters and thus it appears as a horizontal line in top of the figure. Note that to fix the load factor (number of filter bits per Web page) constant at 16 means to change the partition size with the length of the filters. For example, if the length of the filter is 400 bits, then the partition size is 25; if the length of the filter is 800 bits, then the partition size is 50.

The total index size is the sum of both filter size and counter size. Fig. 3 shows the reduction of the index size by using the semi-distributed index sharing mode. Fig.4 shows the reduction of the counter size by using our simplified counter design. The size reduction

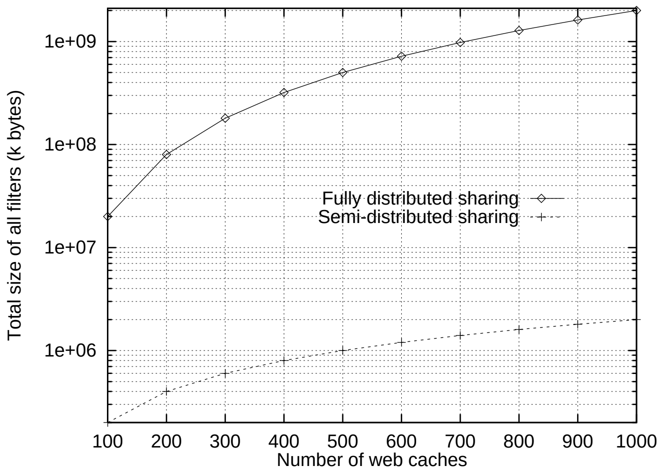

Figure 3: Total filter size difference due to sharing modes.

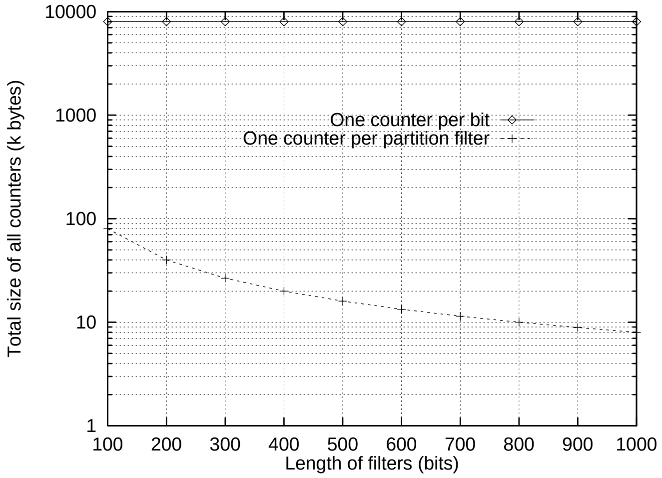

Figure 4: Total counter size difference due to counter designs.

of applying both semi-distributed index sharing mode and simplified counter design is shown in Fig.5.

Note that Fig.3 and Fig.5 are almost identical. This is because under our parameter setting, the shiting from the fully distributed index sharing mode to the semi-distributed index sharing mode for 4,000 Web caches causes orders of magnitude index size reduction while the shifting from one-counter-per-bit design to the one-counter-per-filter design causes counter size reduction of around 80%. If the semi-distributed index sharing mode and the one-counter-per-bit design are used together, the size of all counters will far outweigh the size of all filters. So, when the semi-distributed index sharing mode is used, the total index size with one-counter-per-filter design is about 20% as that with one-counter-per-bit design. In this case, the index size reduction of the one-counter-per-filter design is significant and

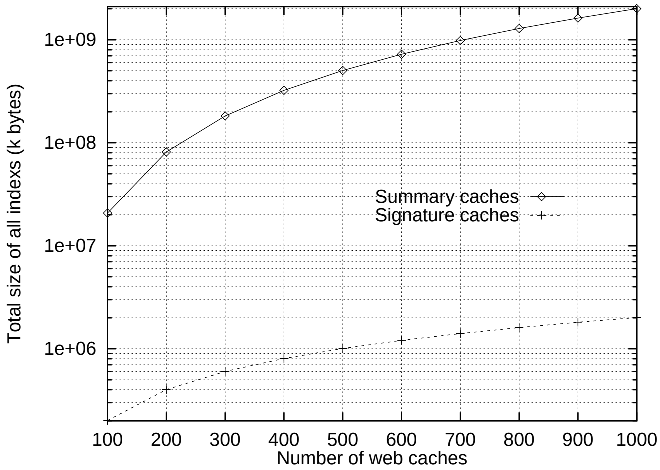

Figure 5: Index size difference between signature and summary cache schemes.

the reduction will be roughly as that shown in Fig.4.

4 Effectiveness of Signature Indexing

Filtering Effect

We analyze the filtering effect of using signatures for Web cache indexing in this section. Our discussion is based on the result of previous researches in signature files [1, 4, 5, 7, 8].

Assume each URL is hashed into a signature of b bits with exactly k bits of 1 and $b-k$ bits of 0 and each partition has m URLs representing m Web pages. A query URL, i.e., the URL of the Web page which is looked for by the query, is also hashed into a query signature of b bits with exactly k bits of 1 and $b-k$ bits of 0. A match of query signature with a partition signature is defined as every bit of 1 is matched by a bit of 1 in the same position of the partition signature.

The probability of false drops, i.e., the probability that a query signature matches with a partition signature but the query URL is not present in the partition, can be computed as follows. For each "1" bit in the query signature, the probability that a signature of a URL in the partition has no "1" bit in this position is $1-k/b$. The probability of all signatures of m URLs in the partition have no "1" bit in this position is $(1-k/b)^m$. So, the probability that at least one "1" bit in all signatures of m URLs in the partition in this position is $1-(1-k/b)^m$. The query signature has k "1" bits. So, the false drop probability for an unsuccessful search is [1]:

$$F = (1-(1-k/b)^m)^k \tag{1}$$

This probability can also accurately approximate the false drop probability for a successful search [2]. From this equation, we derive the desired signature length if we fix k and the false drop probability F:

$$b = \frac{k}{1 - (1 - F^{1/k})^{1/m}} \tag{2}$$

We conducted experiments using a file of 125,000 signatures of length b (b varies from 400 to 1,000) with $k = 4$ "1" bits. The four "1" bits is randomly distributed in b bits. The grouping factor m is fixed at 50. So, there are total of 125,000/50 = 2,500 partition signatures. For each experiment, 20 out of 125,000 signatures are randomly selected as query signatures and are the file of 125,000 signatures is scanned to find true drops. A true drop is a match of a query signature with a partition signature and the partition represented by the partition signature does contain a URL whose signature is the query signature. The match is defined as every "1" bit in the query signature is matched by a "1" bit in the partition signature. The scanning of the file of 125,000 signatures generates true drops and the number of true drops is recorded in this scan. For each query signature, no more than one true drop is recorded for each partition. Each of the 20 query signatures is then checked against 2,500 partition signatures and the number of total drops is recorded. The number of false drops is the difference between the number of total drops and the number of true drops. This experiment is repeated 30 times for each b, using different random seeds. Fig.6 plots the random seeds.

Fig.6 plots the variation of the false drop ratio F with respect to the length of the signatures b. The analytical result is obtained by using Eq.(1) and the experimental result is obtained by experiments described above. The experimental result is slightly better than the analytical result. This is due to the approximate nature of Eq.(1). In Eq.(1), $1-(1-k/b)^m$ is the probability that at least one "1" bit of the partition signature is in the position of a particular "1" bit of the query signature. Because the query signature has k "1" bits, F is approximated as $(1-(1-k/b)^m)^k$. This is an over-estimate because if we already known one "1" bit of the partition signature is in the position of a "1" bit of the query signature, then the probability that at least one "1" bit of the partition signature is in the position of a (second) particular "1" bit of the query signature is between $(1-(1-(k-1)/b))^{m-1}$ and $(1-(1-k/b))^{m-1} * (1-(1-(k-1)/b))$. If $k > 2$, an even smaller probability will be computed for the third "1" bit of the query.

For all practical purposes, we can use Eq.(1) [1] as a simple approximation for the estimation of the filtering effect of signature files.

Obviously, the choice of k affects the value of F for the same b and m pair. The optimal k can be determined as follows. A partition signature is superimposed from m signatures of m URLs. The average number of bits of 1 in the partition signature is:

$$w = b(1 - (1 - k/b)^m) \tag{3}$$

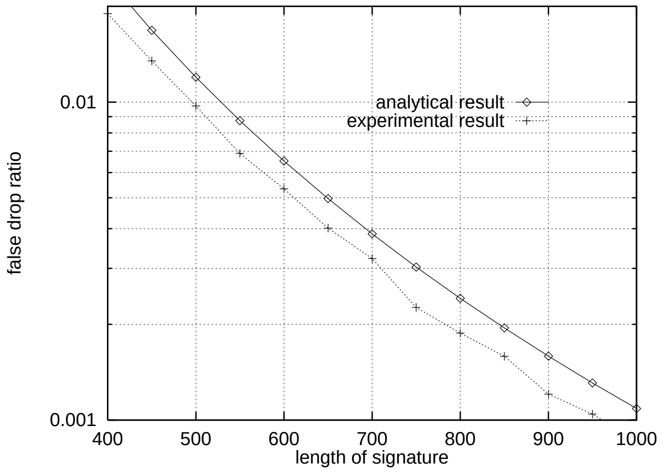

Figure 6: The false drop ratios.

Sacks-Davis [1] shows that the best filtering effect of using signature achieves at approximately $w = b/2$. Substituting $w = b/2$ into Eq.(3), we obtain the optimal k as:

$$k = b(1 - \frac{1}{2^{\frac{1}{m}}}) \tag{4}$$

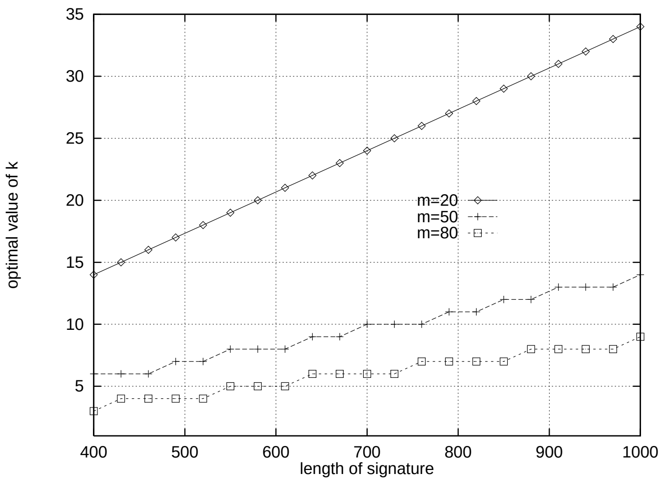

Figure 7: The change of Optimal k values.

Fig.7 plots the variation of the optimal k with respect to b and m by using Eq.(4). Substituting k in Eq.(1) using Eq.(4), we obtain F as the function of b and m, i.e., $F = F(b, m)$ as:

$$F(b, m) = 1/(2^{b(1-1/(2^{\frac{1}{m}}))}) \tag{5}$$

Since b is the length of signatures and m is the size of each partition, load factor (the number of bits per indexed Web page) b/m is proportional to the size of the signature file in the signature caches. For the same Web cache contents, if b/m doubles, then the size of all signature files doubles.

From Fig.6 we see that signatures of length 600 bits is enough to reduce the false drop ratio to 1%. This is sufficient for most practical applications. If the grouping factor $m = 50$, then $(600/8)/50 = 1.5$, the overhead of signatures is only 1.5 bytes per Web page (which averages 8K bytes [9]). This is a very light-weighted index structure.

Search Response Time

In the semi-distributed index sharing mode, if a query for a Web page cannot be satisfied locally in a Web cache, a signal message is sent to the coordinating Web cache to search signature files instead of directly sending signal messages to Web caches that may possibly contain the desired Web page. This means an additional signal message may be sent as compared to that in the fully distributed index sharing mode. However, this additional signal message has limited effect on the network traffic. The total network traffic for answering a query in the fully distributed index sharing mode are:

1. the initial signal message sent by a user to one of the Web caches.

2. in case a cache miss happens, signature files are searched and a list of Web caches are identified as possibly containing the desired Web page. Signal messages are sent to these Web caches.

3. if the Web page is found, a data message is sent to the user who initiated the query. Otherwise, a signal message is sent to the original server of the desired Web page for a fresh copy.

4. a data message is sent from the original server directly to the user who initiated the query.

Since a signal message is usually much smaller than a data message, it is apparent that the additional network traffic caused by using the semi-distributed index sharing mode is no more than one more signal message for answering a query in the fully distributed index sharing mode. Also, the additional signal message that may be sent is between the participating Web caches, which are assumed to be close to each other.

We emphasize that the three improvements of the signature cache scheme: semi-distributed sharing mode, simplified update operation, and simplified design of counters can be used independently. If a decision is made to only adopt simplified update operation and simplified design of counters, then there is no increase in response time, i.e., in this case the signature cache scheme will have the same response time as the summary cache scheme.

5 The Signature Cache Scheme and ICP

The Internet Cache Protocol [16], ICP, is mainly used to exchange hints about the existence of URLs in some neighboring Web caches. So far, ICP has not become an Internet standard. It is obvious that ICP can be improved since ICP protocol sends messages to all participating Web caches in case of a cache miss. The large number of messages cause heavy network traffic among participating Web caches. Both the summary cache scheme [9] and our signature cache scheme are proposed remedies to reduce network traffic overhead.

Fan el al. [9] suggested to add a new opcode, ICP_OP_DIRUPDATE, in ICP version 2 [16] to update the summary index (directory). Our signature cache scheme also has a similar need to update its signature index and similar modification to the ICP is also needed. The main difference between the summary cache scheme (which uses fully distributed index sharing mode) and the signature cache scheme (which may use semi-distributed index sharing) is that if the semi-distributed index sharing mode is adopted, all messages for updating the signature index are sent to the coordinating Web cache.

6 Concluding Remarks

The use of Web caches has become increasingly popular. We recognized that many of the problems and challenges facing the design of large Web caches are similar to those of large databases. Signature method has been studied extensively in the field of text retrieval [2, 3, 4, 6, 8] and recently be proposed to be used in the object-oriented [7] databases. The strength of the signature file is its small size, simple structure, ease of maintenance, and its ability to index text documents. These are precisely what are needed to index large Web caches.

The proposal of summary cache scheme [9] is the first effort to apply signature technique in indexing Web caches. We identified aspects of the summary cache scheme that can be improved and made three improvements: the semi-distributed index sharing mode, simplified update operation, and design of counters. These improvements may be applied independently. We call the resulting scheme the signature cache scheme if all three improvements are applied. With all three improvements, we can achieve orders of magnitude reduction on index size. However, in the worst case the query response time may one additional signal message than that in the summary caches.

If fully distributed index sharing scheme is used and only two of the improvements are applied, namely, the simplified index update operations and the simplified counter design, still, the index size for Web caches can be as small as 20% (assuming 4 bits are used for each counter) of the index size in the summary cache and the response time will be the same as that in the summary cache.

Because of the much smaller index size, the improved schemes (with two or all three of the improvements) have much better scalability than the summary cache scheme. Because of the explosive growth of the World Wide Web, the size of Web caches as well as the size of Web cache indexing structure will also grow accordingly. In this paper, we presented one of the approaches to reduce the Web cache indexing structure for practical Web cache implementations.

References

[1] R. Sacks-Davis, "A Two Level Superimposed Coding Scheme For Partial Match Retrieval," *Information Systems, Vol. 8, No. 4, 1983, pp. 273-280.*

[2] C. Faloutsos, S. Christodoulakis. "Signature files: an access method for documents and its analytical performance evaluation," ACM Transactions on Office Information Systems, Vol. 2, No. 4, Oct. 1984, pp. 267-288.

[3] U. Deppisch, "S-Tree: A Dynamic Balanced Signature Index for Office Retrieval," *Proc of the ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, Sept. 1986, pp. 77-87.*

[4] C. Faloutsos, S. Christodoulakis, "Description and Performance analysis of signature file methods for office filing," *ACM Transactions on Office Information Systems, Vol. 5, No. 3, July 1987, pp. 237-257.*

[5] W. W. Chang, H. J. Schek. "A Signature Access Method for the Starburst Database System," Proc. of the 15th Int'l Conf. on Very Large Data Bases, pp. 145-153, 1989.

[6] C. Faloutsos, "Signature-based text retrieval methods: A survey," *Data Engineering Bulletin, Vol. 13, No. 1, March 1990, pp. 25-32.*

[7] Y. Ishikawa, H. Kitagawa, N. Ohbo, "Evaluation of Signature Files as Set Access Facilities in OODBs," Proc. of the 1993 SIGMOD Conference, Washington, DC, June 1993, pp. 247-256.

[8] D.L. Lee, Y.M. Kim, G. Patel, "Efficient Signature File Methods for Text Retrieval," *IEEE Transactions on Data and Knowledge Engineering, Vol. 7, No. 3, June 1995, pp. 423-435.*

[9] L. Fan, P. Cao, J. Almeida, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *Technical Report CS-TR-98-1361, University of Wisconsin-Madison, 1998. In Proceedings of ACM SIGCOMM'98,* September, 1998.

[10] The Harvest Group. Harvest information discovery and access system. *http://excalibur.usc.edu/,* 1994.

[11] P. B. Danzig, R. S. Hall, M. F. Schwartz, "A case for caching file objects inside internetworks", *Proc. of SIGCOMM'93*, 1993, pp. 239-248.

[12] National Lab for Applied Network Research. Icp working group. *http://ircache.nlanr.net/Cache/ICP/,* 1998.

[13] B. M. Duska, D. Marwood, M. J. Feeley, "The measured access characteristics of world-wide-web client proxy caches", *Proc. of USENIX Symposium on Internet Technology and Systems",* December 1997.

[14] F. Douglis, A. Feldmann, B. Krishnamurthy, J. Mogul, "Rate of Change and other Metrics: a Live Study of the World Wide Web", *USENIX Symposium on Internet Technologies and Systems,* Dec. 1997, pp. 147-158.

[15] The Internet Engineering Task Force, Request for Comments, http://www.rfc-editor.org/rfc.html.

[16] The Internet Engineering Task Force, Request for Comments: 2186, Category: Informational http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2186.txt.