The “Last-Copy” Approach for Distributed Cache Pruning in a Cluster of HTTP Proxies*

Reuven Cohen and Itai Dabran

Technion, Haifa 32000, ISRAEL

Abstract. Web caching has been recognized as an important way to address three main problems in the Internet: network congestion, transmission cost and availability of web servers. As traffic increases, cache clustering becomes a natural way to increase scalability. This paper proposes an efficient scheme for increasing the cache hit-ratio in a loosely-coupled cluster. In such a cluster, each proxy is able to serve every request independently of the other proxies. In order to increase the performance, the proxies may share cacheable content using some inter-cache communication protocol. The main contribution of the proposed scheme is an algorithm that increases the performance (hit-ratio) of any cache-pruning algorithm in such a cluster.

1 Introduction

WWW’s main problems fall into 3 categories: (a) Internet congestion delays; (b) transmission cost, and (c) availability of web servers. Web caching has been recognized as an important way to address these problems. A web proxy cache sits between the Web servers and clients, and stores frequently accessed web objects. The proxy receives requests from the clients and when possible serves these requests using the stored objects. This concept helps the end user, the service provider and the content provider by reducing the server load, alleviating network congestion, reducing bandwidth consumption and reducing network latency.

Cache clustering is a natural way to scale as traffic increases. The idea is to replace a single cache, when it is no longer able to handle all the traffic, with a cluster of caches. However, cache clustering introduces a new “request-routing problem”: which specific cache in the cluster should address each request received from the users. There are several approaches to address this problem, the most popular two are as follows:

1. The URL-space is partitioned, usually through a hash function, across the cluster. Each cache in the cluster is responsible only for the web items whose URLs are hashed to it. The browser, or a proxy in another cluster, hashes the URL, and directs the request message to a specific proxy that is most

* This work was supported by a grant from the Israeli Ministry of Science, Culture and Sport (MOS).

likely to have the required item. The relationship between the caches in such architecture is referred to as *tightly-coupled* [3]. The Cache Array Routing Protocol (CARP) supports this approach.

2. The cluster is arranged in a *loosely-coupled* manner [3]. Each proxy in the cluster is able to serve every request, independently of the other proxies. In order to increase the performance, the proxies may share cacheable content using some inter-cache communication protocol. The most popular protocol for loosely-coupled inter-proxy communication are ICP (Internet Cache Protocol) [14] and HTCP (Hyper Text Caching Protocol) [11].

The main advantages of the tightly-coupled approach over the loosely-coupled approach is that there is no need for inter-cache communication protocol, since a request for a web item is "automatically" routed to the correct proxy. The main advantages of the loosely-coupled approach are attributed to the fact that each proxy server can serve each web item, and therefore (a) this approach can guarantee a much better load balancing; (b) scaling, by adding additional cache proxies, is easy; and (c) this approach is less sensitive to proxy failures.

This paper deals with a loosely-coupled cluster of cache proxies. There are several approaches for "routing" incoming requests. Most of these approaches employ a special logic, referred to as a "director" (or a "dispatcher"), that sits in the entrance of the cluster. The director receives each request, and determines the specific proxy to which this request should be forwarded. Generally, two types of directors are used: a Layer-5 (Application Layer) director and a Layer-4 (Transport Layer) director. The former makes request-routing decisions based on the Application Layer data, like the required URL, cookies, etc., whereas the latter makes these decisions with no awareness to the Application Layer data. The main approach for implementing a Layer-5 director is using another proxy server. Such a proxy terminates the TCP connection with the client, views and analyzes the client request, determines the best proxy machine within the cluster that can serve the request, and forwards the request to this proxy over another TCP connection. In contrast, a Layer-4 director determines the target machine upon receiving the client TCP SYN segment, and therefore without reading the request data. It determines the most available cache proxy, and assigns the new connection to this machine. This usually involves changing the destination IP address of the connection's packets to the internal IP address of that machine through a process known as NAT (Network Address Translation) [9].

Both director types have advantages and disadvantages. The main advantage of a Layer-4 director is that it is much less loaded: unlike a Layer-5 director, that needs to participate in two TCP connections in order to serve each request, the Layer-4 director needs to perform only NAT and IP routing. The main advantage of a Layer-5 director is that it may make smarter request-routing decisions, based on the content of the requests which is unknown to a Layer-4 director. This may help in achieving better load-balancing, and in increasing the cache hit-ratio probability.

A Layer-4 director is the most common approach to route requests to the proxies of a loosely-coupled cluster with no regard to the content of the requests.

However, other approaches exist as well. For instance, another common approach is to use a round-robin DNS in order to send the browser of each user a different IP address for the same cache proxy name.

This paper proposes an efficient scheme for increasing the cache hit-ratio in a loosely-coupled cluster, assuming that the request-routing decision is made by a Layer-4 director, or by any other mechanism that is *unaware of the content of the requests*. Under the considered model a received request is routed to one of the cluster proxies, regardless of the content of the request, either randomly or according to some scheduling policy like round-robin. In a system that employs a Layer-5 director, the director may maintain information regarding the proxy that holds the requested object, in which case the problem is different. The main part of the proposed scheme is an algorithm that increases the performance of any cache pruning algorithm, when the latter is implemented in a cluster of cache servers. A cache pruning algorithm is periodically invoked in order to remove web items from a cache until the cache free space reaches some pre-determined threshold. Cache pruning logic for traditional CPU or I/O subsystems usually makes a decision based on one of the following two parameters:

(1) LRU (Least Recently Used): the last time each item was accessed is considered; the least recently accessed items are assigned a higher removal priority.
(2) LFU (Least Frequently Used): the number of references to each item during the last period of time is considered; the least referenced items are assigned a higher removal priority.

Due to the heterogeneity of the web, the delay variance associated with retrieving a web item from an original server is very large. This gives rise to pruning algorithms that took into account another important parameter, namely

(3) The time it takes to retrieve each item from its original server [10][17]: a web item that can be more rapidly retrieved is assigned a higher removal priority.

However, in a loosely-coupled cluster of proxies, the pruning algorithm executed by each individual proxy should take into consideration another important parameter:

(4) Whether the web item exists or not exists in another proxy within the same cluster. Since we consider a LAN-based, rather than a WAN-based cluster, the cost (in terms of delay and bandwidth) for bringing a web item from another proxy in the same cluster is significantly smaller than the cost for bringing it from the original server. Therefore, items that exist in another proxy within the same cluster should be assigned a higher removal priority.

Designing a pruning algorithm for a loosely-coupled cluster of proxies that takes into consideration all the parameters listed above is difficult for three main reasons:

- The problem of finding the best item to be removed based on two or more of the parameters discussed above is NP-complete [10].

- Estimating the time it takes for bringing a web item from the origin server is a difficult problem, mainly because the load on the servers and on the network may vary rapidly.
- In a loosely-coupled cluster with a Layer-4 director, it is difficult for one proxy to know which web items the other proxies in the cluster hold. A scheme that addresses this problem is presented in [4] and [7]. In this scheme each proxy maintains a compact summary of the web items each other proxy stores. A proxy uses this information in order to identify a local proxy that stores an item for which a request is received. However this approach requires a periodical summary update of each proxy's content, and additional storage capacity at each of the proxies in order to store the cache summaries.

In this paper we address these difficulties using a concept of "Last-Copy" policy, which is enforced by many university libraries. According to this concept, one copy of each important text-book is stamped as "Last-Copy". This copy can be usually used only in the library, in order to increase its availability to someone who really needs it.

We adopt this idea to our scheme in the following way. The cluster may have multiple copies of each web item. However, exactly one copy is marked as "Last-Copy". When the pruning algorithm is executed by each cache proxy, the proxy distinguishes between web items that are marked as "Last-Copy" and web items that are not marked as "Last-Copy". The items from the first group are assigned a much lower pruning priority than the items from the second group, and they are therefore more likely to remain in the cache after the pruning phase terminates. Therefore, regardless of the pruning algorithm employed in the cluster, the concept guarantees with high probability that at least one copy of each item remains in the cluster.

The paper also proposes an ICP-like cache sharing scheme. This scheme ensures that indeed one, and only one, copy of each item is marked as "Last-Copy". The scheme includes a protocol that allows a cluster proxy to search for a web item that is not available in its local cache, and to get a copy of this item from the proxy that holds the "Last-Copy" of this item.

The rest of the paper is organized as follows. In Section 2 we discuss cache pruning algorithms for a single proxy. In Section 3 we discuss this problem in the context of a cluster of proxies. We present our "Last-Copy" scheme and show how this scheme can be integrated into an algorithm for a single proxy in order to work more efficiently in an environment with a cluster of proxies. In Section 4 we present an ICP-like intra-cluster cache sharing protocol, based on the "Last-Copy" concept. In Section 5 we present simulation results that demonstrate the advantages of the proposed scheme, and in Section 6 we conclude the paper.

2 The Cache Pruning Problem

Caching gain can be measured as an Object-Hit-Ratio or as Byte-Hit-Ratio. Object-Hit-Ratio is the ratio of the number of requests served from the cache to the total number of requests. Byte-Hit-Ratio is the ratio of the number of bytes served from the cache to the total number of served bytes. A higher Object-Hit-Ratio indicates higher success in retrieving web items from the cache, whereas a higher Byte-Hit-Ratio indicates a better response to requests of relatively large items. A cache pruning policy that seeks to optimize the Object-Hit-Ratio usually removes large items from the cache, whereas a policy that seeks to optimize the Byte-Hit-Ratio usually does not take into consideration the size of the items. Traditional cache algorithms for CPU or I/O subsystems are based on the concept of "locality of reference", and therefore usually employ an LRU (Least Recently Used) based pruning algorithm. Since CPU or I/O items usually have a fixed size and an equal retrieving time, for such subsystems there is no difference between the Object-Hit-Ratio and the Byte-Hit-Ratio. In contrast, web items have different size and variable retrieving time. Therefore, as explained in Section 1, web cache pruning algorithms take into account not only LRU or LFU considerations, but also the time it takes to obtain the web item from its original web server. Moreover, in order to increase the Object-Hit-Ratio, sometimes on the expense of decreasing the Byte-Hit-Ratio, a pruning algorithm take into account also the size of the page and prune from the cache one big item rather than multiple small items.

In [17] it was found that the performance of a cache pruning algorithm based on the retrieval time only, namely removing the pages whose retrieval time is minimum, is worst than the results achieved by traditional LRU and LFU policies. According to [17], an optimal policy should combine the following 4 parameters: (1) the network latency between the client and the server, which mainly affects the time to open the TCP connection; (2) the bandwidth to the original server; (3) the size of the item; and (4) the popularity of the item, as reflected by LFU. The Hybrid algorithm proposed in [17] keeps a per-origin-server statistics of the estimated connection setup time with this server, and the estimated bandwidth on the path to this server. The cache proxy needs to measure these parameters and to update them each time it accesses the server.

In [15] it is proposed to remove items according to their size only. When the proxy is full, the largest web item is removed. As expected, in [15] and [16] it is argued that this policy indeed maximizes the Object-Hit-Ratio, but has a negative effect on the Byte-Hit-Ratio.

Another work [6] presents the LRV (Lowest Relative Value) policy. This policy combines the parameters of LRU, LFU and object size. According to this algorithm, the removal priority of an item is a function of the time since the last access to this item, the number of accesses to this item during the last Δ seconds, and the size of the item.

A LWU (Least Weighted Usage) removal policy, is presented in [8]. LWU combines only the parameters of LRU and LFU. In experiments using two web server logs, it was found that LWU performs better than LRU and LFU. LRU-MIN [1], is a variant of LRU that tries to minimize the number of items to be replaced. Another LRU variation is LRU-THOLD [1]. This algorithm prevents large web items from entering the cache at all. It was found to perform well for small caches.

3 The “Last-Copy” Concept

In Section 2 we have discussed the most popular cache pruning algorithms for a single proxy. These algorithms are referred in what follows to as *single-proxy (or single-cache) pruning algorithms.* In this section we address the problem of cache pruning in a cluster of cache proxies. Very often, in such a system this issue is addressed by implementing a single proxy pruning algorithm in each of the cluster proxies independently of the other proxies. However, we show that better performance is achieved when the pruning algorithm of each cache takes into account not only the status of the local cache, but also the status of the caches in the other proxies of the cluster. Certainly, in a tightly-coupled system, a global pruning algorithm that takes into account the exact status of each cache can be developed. However, as already noted this paper deals with a loosely-coupled system, where each proxy works independently of the other proxies in the cluster. Moreover, we consider a Layer-4 director. Recall that such a director is unaware of the content of each request, and is therefore unable to synchronize the caches of the various cluster proxies.

In order to take into consideration the content of the other cache proxies in the cluster, we employ the concept of “Last-Copy”. As already indicated, this idea is borrowed from a common policy in university libraries, where one copy of each important textbook is stamped with the “Last-Copy” label. The library imposes a special policy on this copy, in order to improve its availability compared to the other copies of the same book. Our “Last-Copy”-based scheme can be combined with almost every single-proxy pruning algorithm, and in particular with the most common algorithms presented in Section 2. The main idea is as follows. Assume a mechanism that guarantees that exactly one copy of every web item in the cluster is marked as Last-Copy. When the proxy runs its single-proxy pruning algorithm, it assigns a lower removal priority to each local item marked as “Last-Copy”. The purpose is to guarantee that even if an item is pruned from the cache of the other proxies in the cluster, its “Last-Copy” remains.

We now show how this concept can be integrated into the most common single-proxy pruning algorithms that take into consideration the time since the last access to an item (LRU) and the number of accesses to an item (LFU). Each proxy maintains two groups of items, a group of “Last-Copy” items and a group of “Not-Last-Copy” items. In the simple version of the scheme, the proxy only removes items from the “Not-Last-Copy” items list, based on the rules of the single-proxy pruning algorithm. Consequently this approach assigns the “Last-Copy” attribute a definite priority over the attribute(s) of the single-proxy pruning algorithm. An alternative approach, that keeps the temporal locality nature of web access, is to allow the cache to prune a “Last-Copy” item in order to leave a high priority “Not-Last-Copy” item in the cache. A possible way to implement this idea is to determine a threshold on the merit scale of the “Not-Last-Copy” items list, above which items are not discarded from the “Not-Last-Copy” list but from the “Last-Copy” list. As an example, if a single-proxy pruning algorithm is LFU, the algorithm will prune an item from the “Not-Last-Copy” list only if this item was accessed in the last time more than

Δ seconds ago. If no more such items remain in the “Not-Last-Copy” list but more items have to be discarded from the cache, the algorithm starts discarding items from the “Last-Copy” list, provided of course that this list includes old items (that were accessed in the last time more than Δ seconds ago). However, this approach is not examined in this paper.

4 A Distributed Intra-Cluster Scheme based on the “Last-Copy” Concept

In this section we propose a distributed scheme for managing a cluster of caches while maintaining the “Last-Copy” attribute. This scheme guarantees with high probability the following properties:

1. Only one copy of each web item in the cluster is classified as “Last-Copy”. An exception to this property might be if two requests for the same web item are received “almost simultaneously” by two different proxies when a “Last-Copy” for this particular web item does not exist in the cluster. In such a case, both proxies will mark their copy as “Last-Copy”. However, as shown later one of these proxies will re-mark its copy as “Not-Last-Copy” after the copy is requested by a third proxy.
2. If one or more copies of a web item exist in the cache, a request for this item will be served by the local cluster rather than by the original server.

One may consider two models for the management of a cluster with the “Last-Copy” attribute:

1. A centralized model, where all the cluster participants are controlled by a centralized node that guarantees properties 1-2 above.
2. A distributed model, where no centralized controller exists. In such a case the cluster proxies need to run some intra-cluster protocol in order to maintain properties 1-2 above.

Although the implementation of a centralized model is simpler, we concentrate in this paper on the distributed approach. This is because the considered loosely-coupled system with a Layer-4 director does not scale well with a centralized controller that may easily become a bottleneck of the cluster. The work such a centralized controller has to perform, is much heavier than the “standard” work performed by a “regular” Layer-4 director.

Recall that in the considered system the proxy servers are connected to each other through a high-speed LAN, like 1Gb/s switched-Ethernet. Therefore IP multicast can be efficiently implemented within the cluster. The proposed scheme is based on the ICP (Internet Caching Protocol) as defined in [14]. ICP is a lightweight protocol, used for inter-proxy communication. The proxies exchange ICP query and reply messages in order to select the most appropriate neighboring proxy from which a requested object can be retrieved. A cache proxy sends an ICP query message, using either unicast or multicast, over UDP to its neighbors, and gets back from each neighbor a HIT or a MISS reply. The proxy analyzes the received reply messages and determines how to proceed accordingly. Related issues, like when to send an ICP query, to which cache proxy should an ICP query be sent, and how to process the received replies, are out of the scope of ICP.

In the proposed scheme, when a proxy receives a request for a web item that is not found in the local cache, it multicasts an ICP query message to the group of proxies in the cluster. One important advantage of our scheme over schemes that use ICP with multicast [12,13] is that even if the searched item is found by many proxies, only one reply is sent back to the searching proxy. This reply is sent by the holder of the copy marked as "Last-Copy". Note that by minimizing the number of responses the searching proxy receives, we reduce the processing power this proxy needs to spend for opening and discarding irrelevant responses

Suppose that a proxy server p_i receives a user request for a web item w. The following cases are possible:

1. w exists in the local cache and it is valid.
2. w exists in the local cache but it is not valid.
3. w does not exist in the local cache.

Note: The proposed scheme is orthogonal to the validation model employed between the cluster proxies and the original web servers.
The response of p_i to the user request is as follows:

1. If w exists in the local cache and it is valid, it is sent to the requesting user.
2. If w exists in the local cache but it is not valid then:
   (a) If $p_i(w)$, namely the copy of w at p_i, is marked as "Last-Copy", then p_i checks the validity of $p_i(w)$ against the original server. If $p_i(w)$ is not fresh, p_i gets a fresh copy of w from the original server and sends it to the requesting user. The status of the fresh copy remains "Last-Copy".
   (b) If $p_i(w)$ is not marked as "Last-Copy", then w is requested from the local proxy that maintains the "Last-Copy" of w. Let this proxy be p_j. Proxy p_i discovers p_j by multicasting a special ICP query message called "Last-Copy-Search". The protocol then continues as follows:
      i. If $p_j(w)$ is valid, then p_j sends w to p_i, and p_i sends it to the user.
      ii. If $p_j(w)$ is not valid, then p_j informs p_i by unicast that the "Last-Copy" is not valid. In addition p_j marks its copy as "not-Last-Copy". Proxy p_i requests a fresh copy to be cached from the original server, marks it as "Last-Copy", and sends it to the user.
      iii. If no reply from a "Last-Copy" owner of w is received within a time-out period, p_i requests a fresh copy to be cached from the original server, marks this copy as "Last-Copy", and sends it to the requesting user.
3. If w does not exist in the local cache, p_i sends a multicast "Last-Copy-Search" and continues as in step 2(b) above.

According to the scheme described so far, the proxy that marks its copy of some web item as "Last-Copy", is the proxy that has received this item from the origin server rather than from another proxy. Nevertheless, due to a premature timeout or to ICP message loss, it is possible that two or more copies of the same item are marked as "Last-Copy" at the same time. Suppose, for example, that a proxy p_i searches for an item w whose "Last-Copy" version exists at proxy p_j. If p_j does not respond before p_i times out, or if the response message of p_j is lost, p_i will bring a copy of w from the original server and mark it as "Last-Copy".

However, the existence of two or more copies marked as "Last-Copy" does not affect the correctness of the whole scheme. Moreover, when a "Last-Copy-Search" message for this copy is multicast in the next time, this problem is reduced in the following way. Let the searching proxy be p_i, the responding proxies be p_j and p_k, and the web item be w. Let the IP addresses of these proxies be $Addr(p_i)$, $Addr(p_j)$ and $Addr(p_k)$ respectively. Note that p_i knows $Addr(p_j)$ and $Addr(p_k)$. If $Addr(p_j) < Addr(p_k)$, then p_i requests w from p_j and vice versa. The usage merit (LFU, LRU etc.) of w in p_k will be decreased, and it will eventually be pruned.

5 Simulation Results

In order to evaluate the performance improvement of the proposed scheme, we modeled a distributed proxy cluster managed by a Layer-4 director. The director uses a simple round-robin scheduling algorithm in order to assign incoming requests to the cluster proxies. We selected the two most popular single-proxy pruning algorithms, LRU and LFU, and evaluated the contribution of the "Last-Copy" scheme by comparing their performance, in terms of Object-Hit-Ratio and Byte-Hit-Ratio, with and without the "Last-Copy" scheme. When the "Last-Copy" scheme is not implemented, a proxy returns the requested web item if it exists in its local cache, and sends a query message to the cluster participants if the item does not exist. If no reply from one of the other proxies is received within a time-out period, the proxy requests a fresh copy from the original server, and returns it to the requesting user. If one or more replies are received, the proxy requests the web item from the first proxy to respond and forwards it to the requesting user. When the "Last-Copy" scheme is implemented, the proxy algorithm is as described in Section 4. In both cases a request received by a proxy is a "hit" if the requested object exists *somewhere* in the cluster. It is a "miss" if it does not exist anywhere in the cluster.

We used a ClarkNet-HTTP trace in order to test the behavior of the cluster. ClarkNet is a full Internet access provider for the Metro Baltimore-Washington DC area. The trace contains one-week worth of all HTTP requests[1]. Each request is associated with a timestamp taken from a clock with a granularity of 1 second. We created from this trace an input file containing only the successful

[1] The logs were collected by Stephen Balbach of ClarkNet, and contributed by Martin Arlitt and Carey Williamson of the University of Saskatchewan.

transactions, namely transactions to which an HTTP "200 OK" response is received. According to [2], 88.8% of the transactions (about 1,400,000) were found to be successful. Figure 1 depicts the size distribution of the items in the trace. It is evident that most of the items are relatively small (less than 10KB). In order to validate our simulation model, we started with a 1-proxy cluster that uses LRU pruning without the "Last-Copy" scheme, and found the results to be very similar to those reported in [5] for NLANR trace driven simulation.

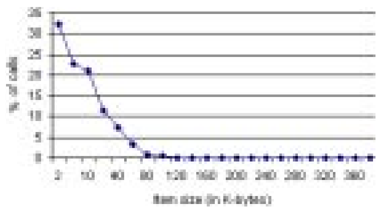

Fig. 1. Calls distribution according to item size

Figure 2 depicts the results achieved for a cluster of 15 proxies when LFU is used as the single-proxy pruning algorithm. In Fig. 2(a) we show the Object-Hit-Ratio, whereas in Fig. 2(b) we show the Byte-Hit-Ratio. The X-axis indicates the relative cache size of the *whole cluster*, namely the ratio between the cluster capacity and the total size of all the requested items. We see that the "Last-Copy" scheme increases the Object-Hit-Ratio and the Byte-Hit-Ratio substantially. It is interesting to note that the relative contribution of the scheme is only slightly affected by the cache size. We see an increase of 45% (from 55% to 80%) in the Object-Hit-Ratio when the relative cache size is 7%, and an increase of 40% when the relative cache size of the whole cluster is 20%.

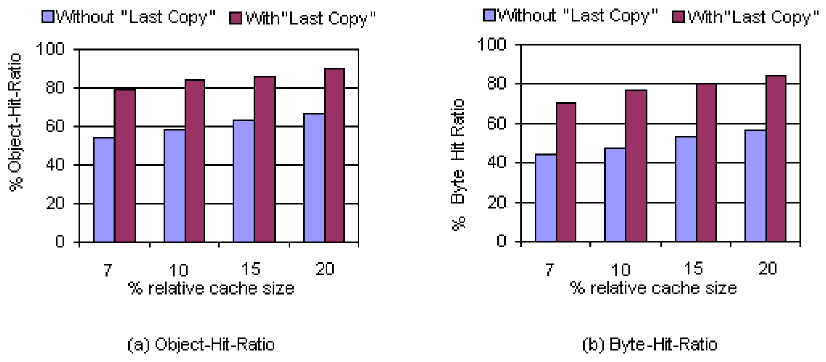

Fig. 2. A cluster of 15 proxies using LFU

Figure 3 depicts the results achieved for a cluster of 15 proxies when LRU is used as the single-proxy pruning algorithm. Generally the Byte-Hit-Ratio increases when large items are requested very often. Since in the considered trace, that reflects the situation in the Internet today, most of the popular items are relatively small (Fig. 1), the Byte-Hit-Ratio achieved for both LFU and LRU (Fig. 2(b) and Fig. 3(b) respectively) is smaller than the Object-Hit-Ratio (Fig. 2(a) and Fig. 3(a)). However, since the “Last-Copy” scheme increases the availability of less popular items, namely larger items in our trace, it is evident that for both LFU and LRU the contribution of “Last-Copy” to the Byte-Hit-Ratio is higher than to the Object-Hit-Ratio. For instance when the cache relative size is 7% the contribution of the “Last-Copy” scheme with LFU is 45% to the Object-Hit-Ratio and 55% to the Byte-Hit-Ratio. Similarly, for LRU the numbers are 14% and 20% respectively. However, the most important conclusion one may derive from Fig. 2 and Fig. 3 is as follows. Without the “Last-Copy” scheme, the Object-Hit-Ratio and Byte-Hit-Ratio performance of LRU are much better than those of LFU. With the “Last-Copy” scheme the differences almost disappear: LFU performs slightly better than LRU with regard to the Object-Hit-Ratio and the Byte-Hit-Ratio. This implies that the influence of LFU as a part of sophisticated web caching pruning algorithms, such as the Hybrid Algorithm mentioned in [17] and the LRV algorithm mentioned in [6], can achieve better Object-Hit-Ratio and the Byte-Hit-Ratio results in a cluster of proxies, when the “Last-Copy” scheme is employed.

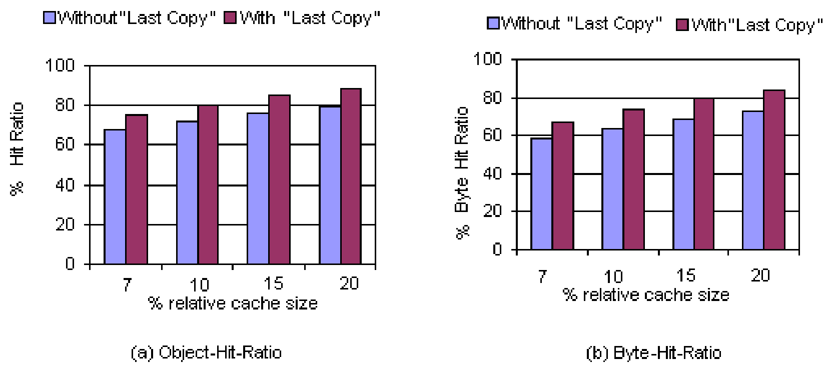

Fig. 3. A cluster of 15 proxies using LRU

Figure 4 depicts the results achieved for various cluster sizes when LFU is used as the single-proxy pruning algorithm. Figure 4(a) shows the Object-Hit-Ratio improvement while Fig. 4(b) shows the Byte-Hit-Ratio improvement. As expected, for all cache sizes the improvement increases with the size of the cluster, because less non “Last-Copy” items are kept. For instance with a relative cache size of 15% the Object-Hit-Ratio improvement when the cluster has 5 proxies is 20%, while with 40 proxies it is 55%. Similarly the Byte-Hit-Ratio increases by 28% and 80% respectively. It is also evident that when the cluster is "relatively large", namely contains more than 10 proxies, the contribution of the "Last-Copy" scheme decreases as the cache size increases.

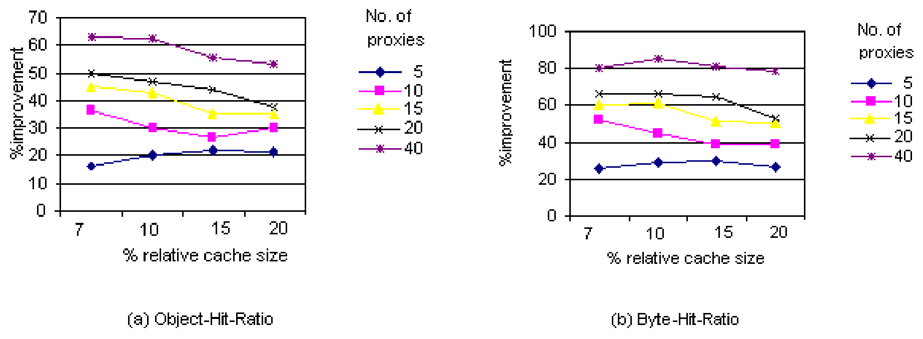

Fig. 4. LFU improvement as a function of relative cache size and number of proxies

Figure 5 depicts the Object-Hit-Ratio and the Byte-Hit-Ratio as a function of the number of proxies in the cluster, when the relative cache size is 20% and LFU is used as the single-proxy pruning algorithm. The most interesting conclusion from these graphs is that without the "Last-Copy" scheme the Hit-Ratio decreases when the size of the cluster increases, whereas with the "Last-Copy" scheme the Hit-Ratio remains constant.

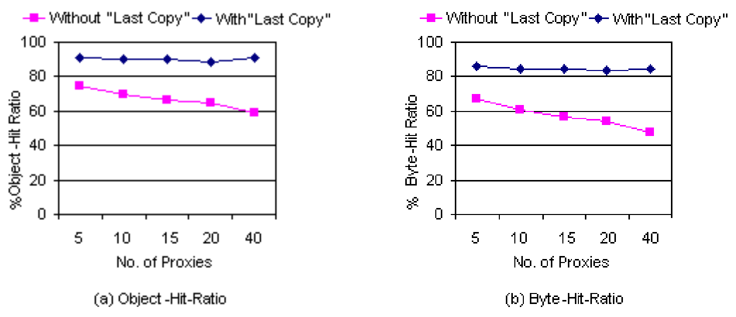

Fig. 5. The Results for LFU when the relative cache size of the whole cluster is 20%

The same conclusion can be derived for LRU (Fig. 6). The explanation to this is as follows. Since we consider a "miss" each request that does not exist in the whole cluster, when the number of proxies increases and the "Last-Copy" scheme is not employed, there is a high probability for a popular item to be stored at multiple caches. This decreases, of course, the total number of items that can be cached in the whole cluster, and therefore reduces the Hit-Ratio. When the “Last-Copy” scheme is implemented with LFU or LRU, a lower pruning probability is assigned to items that do not exist in other caches of the cluster. Therefore the number of items stored in the whole cluster is not affected by the number of proxies in the cluster and the Hit-Ratio remains constant.

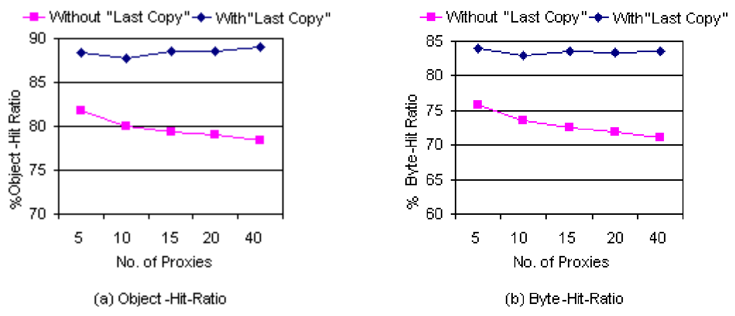

Fig. 6. The Results for LRU when the relative cache size of the whole cluster is 20%

In the discussion so far we considered a “hit” each request for an object/byte that exists in any proxy of the cluster. However, in order to investigate a possible drawback of the proposed “Last-Copy” scheme, we need to distinguish between a “Local-Hit-Ratio” and a “Cluster-Hit-Ratio”. The former term applies to the case where the request received by a proxy can be fulfilled using the proxy local cache only, whereas the latter term applies to the case where the request can be fulfilled using all the cluster caches. Each request that is a “Cluster-Hit” but not a “Local-Hit” requires the transfer of the object between two cluster proxies, and therefore increases the load on the cluster. Figure 7 depicts the “Local-Object-Hit-Ratio” of LRU and LFU for a cluster of 15 proxies. As expected, the “Local-Hit-Ratio” is lower when the “Last-Copy” scheme is employed. This is because the simulated scheme assigns a lower priority to “non-Last-Copy” items even if their LRU/LFU priority is high.

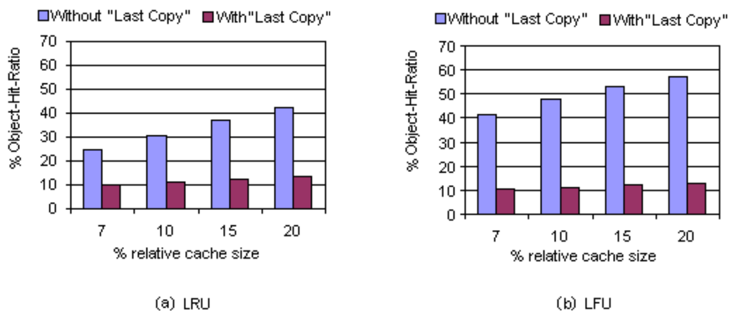

Fig. 7. “Local-Hit-Ratio” of a 15-proxy cluster with LRU and LFU

However, despite of the lower "Local-Hit-Ratio" of the "Last-Copy" scheme, the "Last-Copy"-based approach does not necessarily increase the processing load of the cluster proxies. Recall that when the "Last-Copy" approach is implemented, each ICP search message is responded by at most one proxy - the one that holds the "Last-Copy" version of the requested item. In contrast, when the "Last-Copy" approach is not implemented, each search message is responded by all the proxies of the cluster [12]. Hence, much more control messages have to be sent and processed. Figure 8 shows the number of control (ICP search and reply) messages sent with and without the "Last-Copy" scheme for a cluster of 15 proxies when LRU and LFU are used. Since according to Fig. 7 the "Local-Hit-Ratio" of both LFU and LRU while using the "Last-Copy" scheme is about 12%, the number of internal communication messages is also almost fixed: about 1.6 requests per message. When our scheme is not implemented, the "Local-Hit-Ratio" increases, the number of query control messages decreases, but since each proxy responds to every request message, the number of internal control messages per request increases substantially to 9-11 for LRU and 6.5-9 for LFU.

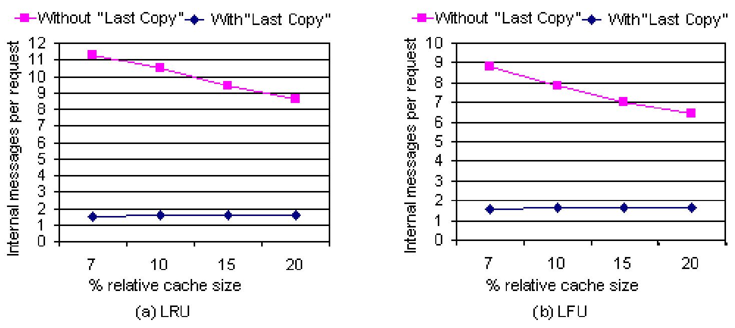

Fig. 8. Number of internal messages per request

6 Conclusions

In this paper we proposed an efficient scheme for increasing the cache hit-ratio in a loosely-coupled cluster, where proxies share cacheable content using some inter-cache communication protocol. The main part of the proposed scheme is an algorithm that increases the performance of any cache pruning algorithm when the latter is implemented in a cluster of cache servers. The solution presented in this paper is based on the concept of "Last-Copy" policy. According to this policy, exactly one copy of each web item in the cluster is marked as "Last-Copy". Items marked as "Last-Copy" are assigned a much lower pruning priority than "non-Last-Copy" items. This concept increases the number of web items available at the cluster, and therefore increases the cache hit ratios. The paper proposed an ICP-based cache sharing scheme that ensures that indeed only one copy of each item is marked as “Last-Copy”.

In order to evaluate the performance improvement of the proposed scheme, we modeled a distributed proxy cluster managed by a Layer-4 director. The director employs a simple round-robin scheduling algorithm in order to assign incoming requests to the cluster proxies. We evaluated the contribution of the “Last-Copy” scheme by investigating the performance of the two most popular single-proxy pruning algorithms: LFU and LRU, with and without the “Last-Copy” scheme.

The simulations show that the “Last-Copy” scheme increases the Object-Hit-Ratio and the Byte-Hit-Ratio substantially. The relative contribution of the scheme is only slightly affected by the cache size. Since the scheme increases the availability of less popular items, namely larger items in our trace, it is evident that for both LFU and LRU the contribution of “Last-Copy” to the Byte-Hit-Ratio is higher than its contribution to the Object-Hit-Ratio. Another interesting result is that without the “Last-Copy” scheme, the Object-Hit-Ratio and Byte-Hit-Ratio of LRU are much better than for LFU. However with the “Last-Copy” scheme the differences disappear and both algorithms perform very similarly.

Another important advantage of the proposed scheme is that it decreases the internal communication substantially. This is because only the owner of the “Last-Copy” item responds to each request, rather than all the proxies when the “Last-Copy” scheme is not implemented.

References

1. M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox. Caching proxies: Limitations and Potentials. In *1995 World Wide Web Conference*, 1995.
2. M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In *ACM SIGMETRICS*, Philadelphia, PA, USA, Apr. 1996.
3. I. Cooper, I. Melve, and G. Tomlinson. Internet Web Replication and Caching Taxonomy. RFC-3040, Jan. 2001.
4. L. Fan, P. Cao, , and J. A. A. Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *ACM SIGCOMM*, Vancouver, Canada, 1998.
5. K. Psounis and B. Prabhakar. A Randomized Web-Cache Replacement Scheme. In *IEEE Infocom 2001 Conference*, Apr. 2001.
6. L. Rizzo and L. Vicisano. Replacement Policies for a proxy cache. Technical Report RN/98/13, University College London, Department of Computer Science, Feb. 1998.
7. Rousskov and D. Wessels. Cache Digests. In *3rd International WWW Caching Workshop*, June 1998.
8. Y. Shi, E. Watson, and Y. Chen. Model-Driven simulation of world wide web cache policies. In *Winter Simulation Conference*, Dec. 1997.
9. P. Srisuresh and D. Gan. Load Sharing using IP Network Address Translation (LSNAT). RFC-2391, Aug. 1998.
10. R. Tewari, H. M. Vin, Asit, and D. Sitaramy. Resource-based Caching for Web Servers. In *SPIE/ACM Conference on Multimedia Computing and Networking*, Jan. 1998.

11. P. Vixie and D. Wessels. Hyper Text Caching Protocol (HTCP/0.0). RFC-2756, Jan. 2000.
12. D. Wessels and K. Claffy. Application of the Internet Cache Protocol (ICP). RFC-2187, Sept. 1997.
13. D. Wessels and K. Claffy. ICP and the Squid Web Cache, Aug. 1997.
14. D. Wessels and K. Claffy. Internet Cache Protocol (ICP). RFC-2186, Sept. 1997.
15. S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. Removal Policies in Network Caches for World-Wide Web Documents. In *SIGCOMM*, 1996.
16. S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. Errata for Removal Policies in Network Caches for World-Wide Web Documents, Feb. 1997.
17. R. P. Wooster and M. Abrams. Proxy Caching that estimates page load delays. In *6th International World-Wide Web Conference*, Santa Clara, California, USA, 1997.