Cooperative TransCaching: A System of Distributed Proxy Servers for Web Content Adaptation

Claudia Canali
University of Parma
claudia@weblab.ing.unimo.it

Valeria Cardellini
University of Roma Tor Vergata
cardellini@ing.uniroma2.it

Michele Colajanni
University of Modena
colajanni@unimo.it

Riccardo Lancellotti
University of Roma Tor Vergata
riccardo@weblab.ing.unimo.it

Philip S. Yu
IBM T.J. Watson Research Center
psyu@us.ibm.com

1. INTRODUCTION

The Web is rapidly evolving towards a highly heterogeneous accessed environment, due to the variety of new devices with diverse processing, storage, display capabilities, and network interfaces that are gaining access to the Internet. Studies have shown that in a few years these emerging consumer devices will be the predominant fraction of Internet clients. Hence, there is an increasing demand for solutions that enable the transformation of Web content for adapting and delivering it to diverse destination devices.

Tailoring Web content to match the specific device capabilities requires functionalities for content adaptation, namely *transcoding*. These transcoding services are typically carried out by the content Web server or by some edge proxy server. In the latter approach (e.g., [3, 4]), a proxy server, located in an intermediate position in the delivery chain between the client device and the content server, fetches and transcodes the desired content on-the-fly, before delivering the result to the client. The proxy server can cache the result of the transcoding, thus avoiding round-trips to the content server and transcoding operations when transcodable content can be served from the cache [4]. So far, the proxy-based approach to content adaptation is typically carried out by some edge proxy server that directly connects to the clients.

As the proxy-based approach ma be subject to a limited scalability [3], we believe that alternative solutions are necessary to improve the user response time. In this paper, we propose a new alternative that considers distributed systems of cooperative proxy servers which collaborate in discovering, transcoding, and delivering multiple versions of Web objects. The goals are to reduce the user response time and bound its variability by exploiting the benefits deriving from resource discovery and adaptation through other proxy servers.

The presence of multi-version content in the proxy caches augments the traditional cache cooperation issues, because we have to consider the possibility of recognizing, discovering, and caching multiple variants of the same resource. We investigate different schemes for cooperative proxy caching and transcoding that can be implemented in the existing Web infrastructure and compare their performance through prototypes that extend Squid operations to an heterogeneous client environment. The first extension enhances the traditional cache server, and transforms it into an active smart intermediary server that not only caches Web objects but also transcodes them and stores the results [4]. The second extension allows multi-version content discovery through cooperative proxy servers that can be organized in hierarchical and flat topologies, and can use query- and summary-based cooperation protocols.

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

2. COOPERATIVE CACHING AND TRANSCODING

We consider that proxy cooperation can be established along a vertical direction, namely hierarchical Web caching, or along an horizontal direction, namely flat or distributed Web caching [5]. To describe the novel cooperation schemes, we show in Figure 1 the case of a flat architecture. Similar considerations hold for a hierarchical-based architecture.

We identify three main phases that may require some cooperation among the active proxies, namely *discovery* (steps 2, 4, and 5 in Figure 1), *transcoding* (when the transcoder component is involved in steps 3b and 6b), and *delivery* phases (steps 6 and 7).

During the *discovery* phase, the active proxies may cooperate to discover the version of the Web object requested by the client. Since many versions of the same object typically exist in the caches, in this phase it is necessary to carry out a multi-version lookup process that may require cooperation among the active proxies. The discovery phase includes a local lookup and may include an external lookup whose execution mode depends on the proxy cooperation model. There are three possible results of the multi-version lookup process (both local and external). (1) *Exact hit*: the cache contains the exact version of the object, that can be immediately delivered to the client (in the case of local lookup) or to the requesting proxy (in the case of external lookup). (2) *Useful hit*: the proxy cache contains a more detailed and transcodable version of the object that can be transformed to meet the client request. (3) *Miss*: the proxy cache does not contain any valid copy that can satisfy the client request. When a global cache miss occurs, the client request is forwarded to the content server. (We assume that the Web server returns the original version of the object to the requesting proxy, without performing any transcoding operation.)

The *transcoding* phase is necessary when either a global miss or a useful hit occurs. Any proxy of the considered cooperative system is assumed to be equipped with suitable software that can perform the transcoding operations required by different client devices.

Once an exact version of the requested object is retrieved, the

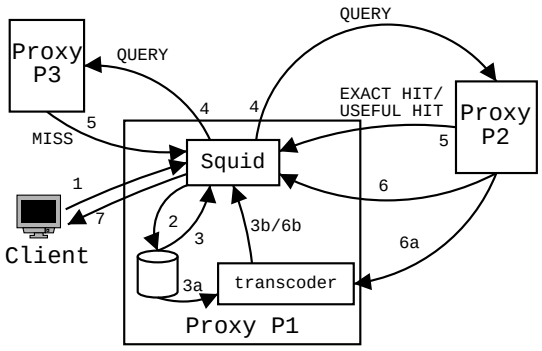

Figure 1: Operational example in a flat query-based architecture.

delivery phase transfers the resource to the client. The final delivery is carried out by the edge proxy that is contacted by the client. Hence, if the object is found in another proxy, the delivery phase includes the object transmission to the edge proxy.

Figure 1 shows how a client request is serviced in the version of our prototype using a query-based cooperation protocol in a distributed flat architecture (the case of summary-based cooperation is conceptually similar). In the example, the architecture consists of three proxies (P_1, P_2 and P_3), where P_1 receiving the client request is shown with more details. When P_1 accepts the request (step 1), it performs a multi-version lookup in its local cache (step 2). If there is an exact version of the requested resource (local exact hit), it is retrieved from the cache (step 3) and sent to the client (step 7). If there is a useful version (local useful hit), the resource is first adapted by the transcoder component (steps 3a, 3b) and then sent to the client (step 7). A local miss event activates the cooperative multi-version lookup process, consisting in a query phase (step 4) and a reply phase (step 5). There are three types of replies: in the case of exact remote hit, the resource is fetched from the remote cache server (step 6); in the case of useful hit, the resource is retrieved from the peer (step 6a) and transcoded (step 6b) before the transmission to the client; a global miss event (not shown) requires that the resource is retrieved from the content Web server and adapted to meet the client needs.

We implemented a specific prototype for each cooperative architecture. The basic software platform for all prototypes is the Squid Web proxy cache, version 2.4 [6]. The main novelties include the management of caching and lookup (both local and remote) of multi-version content and provisioning of transcoding functionalities to adapt the requested resource to the client device capabilities. A technical description of hierarchical- and flat-based prototypes can be found in [2]. Let us give some additional details about flat topologies where all nodes are peers. In these architectures we investigate how to support multi-version lookup through two main classes of protocols: *query-based* and *summary-based* (being the latter a simplified version of the directory-based protocols).

Query-based protocols are conceptually simple. When a cache server experiences a local miss, it sends a query message to all its peers to discover whether one of them caches a valid copy of the requested resource. In the positive case, the recipient proxy server replies with an exact hit message or with a useful hit response, otherwise it may reply with a miss message or not reply at all. In the case of useful hit, the response message provides some information about the available version of the resource to allow its retrieval. We chose ICP as the query-based cooperation protocol and included the support for multiple-version lookup to the Squid version of ICP.

For the summary-based protocols we consider a distributed scheme, in which each proxy keeps a directory of the resources that are cached in the other peers, and uses the directory as a filter to reduce the number of queries. We chose Cache Digests as the representative of the summary-based cooperation scheme. We added the support for multiple versions to the summary-based lookup process in a transparent way by URL-encoding the resource version identifier. The basic mechanism of Cache Digests cooperation is preserved. However, the lookup process becomes more expensive because a lookup for every possible useful version must be performed.

3. SUMMARY OF RESULTS

All prototypes were tested extensively to study the performance impact of different cooperation schemes for discovery and transcoding. Experiments demonstrate that all proposed cooperative schemes are immediately applicable to the existing Web infrastructure. Moreover, we have found that cooperative schemes can substantially reduce the system response time compared to the non-cooperative case, in which the proxies do not collaborate in the multi-version resource discovery and transcoding processes. Under a realistic workload model and with the Web servers located remotely with respect to the clients and proxies, the median of the system response time for ICP is 130 msec against 504 msec of the no-cooperation scheme, while the 70-percentile are 335 and 1115 msec, respectively (the corresponding global cache hit rates are 58.4% and 10.7%).

Our results clearly demonstrate the advantages of cooperative caching and transcoding through flat topologies over a pure hierarchical cooperation scheme. This result confirms the intuition that scalability, bottlenecks and coverage problems shown by the pure hierarchical architectures are worsen by the fact that active proxies are not simple object repositories, but they perform also transcoding operations.

Among the flat architectures, a query-based protocol, such as ICP, seems to offer the best response times due to its cache hit rate higher than that of summary-based protocols that are subjects to false misses and false hits. However, a summary-based protocol, such as Cache Digests, can provide best performance when the system is subject to heavier workload scenarios and/or inter-proxy connections are highly loaded. Even in those instances, Cache Digests experiences a very low cooperation cost, whereas the high cooperation overhead of ICP impacts negatively on the response time much more than its higher cache hit rates.

A plethora of research issues can be studied for the topic of proxy systems that cooperate in multi-version content caching, discovery, and transcoding. Major details about our research activities can be found in [1]

4. REFERENCES

[1] C. Canali, V. Cardellini, M. Colajanni, and R. Lancellotti. Proxy servers for cooperative caching and transcoding, 2003. `http://weblab.ing.unimo.it/research/trans_caching.shtml`.

[2] C. Canali, V. Cardellini, and R. Lancellotti. Squid-based proxy server for content adaptation. Technical Report TR-2003-03, Dept. of Computer Engineering, Univ. of Roma "Tor Vergata", Jan. 2003.

[3] W. Y. Lum and F. C. M. Lau. On balancing between transcoding overhead and spatial consumption in content adaptation. In *Proc. of ACM Mobicom 2002*, pages 239–250, Sept. 2002.

[4] A. Maheshwari, A. Sharma, K. Ramamritham, and P. Shenoy. TransSquid: Transcoding and caching proxy for heterogeneous e-commerce environments. In *Proc. of 12th IEEE Int'l Workshop on Research Issues in Data Engineering*, pages 50–59, Feb. 2002.

[5] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[6] Squid Internet Object Cache. `http://www.squid-cache.org`.