A Distributed Architecture of Edge Proxy Servers for Cooperative Transcoding

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

Abstract

The large variety of devices that are gaining access to the Internet requires novel server functionalities to tailor Web content at run-time, namely transcoding. Traditional schemes assign transcoding operations to the Web server or single edge proxies. We propose an alternative architecture consisting of cooperative proxy servers which collaborate in discovering and transcoding multiple versions of Web objects. The transcoding functionality opens an entirely new space of investigation in the research area of cache cooperation, because it transforms the proxy servers from content repositories into pro-active network elements providing computation and adaptive delivery. We investigate and evaluate experimentally different schemes for cooperative discovery of multi-version content and transcoding in the context of a flat topology of edge servers.

1. Introduction

The emerging Web-connected devices, such as handheld computers, personal digital assistants (PDAs), mobile phones, and other pervasive computing devices, differ considerably in network connectivity, processing power, storage, display, and format handling capabilities. Hence, there is a growing demand for solutions that enable the transformation of Web content for adapting and delivering it to different devices.

The process of converting a multimedia resource from one form to another, called *transcoding*, can be applied to transformations within media types and/or across media types. The existing solutions to deploy Web content adaptation fall into three broad categories depending on the system component that performs the adaptation process [1, 7]: *client-based*, *proxy-based*, and *server-based* adaptation.

The proxy-based approach seems the most viable solution or, at least, is the most investigated in literature (e.g. [5, 6, 7, 8]), because it allows to shift the load from content-providing Web servers and to simplify their design. Here, some edge server is enhanced with functionalities to transcode the content on-the-fly, before delivering it to the client. As this server can cache original and transcoded objects, it may save round-trip times to reach the content Web server and/or CPU cycles when the requested object is found in the cache. As transcoding operations can be computationally expensive, the proxy-based approach may be subject to limited scalability issues [7], that Fox et al. address by proposing a cluster of locally distributed proxies [5]. This approach may solve the CPU-resource constraint, but it tends to move the system bottleneck from the proxy CPU to the interconnection of the cluster.

In this paper, we explore a quite different alternative that considers distributed systems of cooperative edge servers which collaborate for the discovery and transcoding of Web content. The goals are to face more efficiently the heterogeneity of the client environment by reducing the user response time and bounding its large variability through three possible benefits provided by a cooperative architecture: avoidance of the network bottleneck of a cluster-based architecture; higher scalability because computational costs of transcoding can be shared among multiple nodes; reducing the amount of necessary transcoding operations thanks to the increased cache hit rate due to cooperation. In this paper, we focus on cooperative resource discovery schemes that operate in a flat architecture of peer-to-peer edge servers. We compare their performance through a Squid-based prototype.

Provisioning services for transcoding opens an entirely new space of investigation in the research area of cache cooperation, because it transforms the proxy servers from content repositories into pro-active intermediaries provid-

ing adaptive computation and context-aware delivery [8]. Moreover, the presence of multi-version content in the proxy caches requires us to address various issues related to recognizing, discovering, and caching of the variants of the same resource that are obtained by transcoding operations. We are not aware of any other research work dealing with a prototype implementation of cooperative transcoding and caching proxy systems with flat peer-to-peer topologies through query- and summary-based cooperation protocols. Some preliminary results have been obtained by the authors through simulations in a context of hierarchical architectures [3]. We demonstrate that all proposed cooperative schemes are immediately applicable to the Web infrastructure. The real test-beds allow us to evaluate the reduction of the user response time achievable by cooperative discovery and transcoding schemes with respect to the case of a system of non-cooperative proxies.

The rest of this paper is organized as following. Section 2 discusses the requirements and mechanisms of a system for cooperative caching and transcoding. Section 3 explores discovery topologies and protocols. Section 4 presents the experimental results. Section 5 concludes the paper with some final remarks.

2. Distributed system of transcoding proxies

The features of client devices vary widely in screen size and colors, processing power, user interface, software, and network connections. The edge proxy receiving a request for a transcoded resource has to fit the client capability needs as much as possible. An object in the proxy cache which has been previously transcoded may be further transcoded to yield a lower quality object. Different versions of the same object (and the transcoding operations among them) can be represented through a *transcoding relation graph* [3].

Cooperation for discovery and transcoding in a distributed system can be established along a vertical direction (e.g., hierarchical Web caching) or along an horizontal direction (e.g., peer-to-peer or flat distributed Web caching) [9, 10]. The architecture we consider in this paper is based on a flat topology of edge servers that are contacted directly from the clients. In a distributed system for caching and transcoding, we identify three main phases that may require some cooperation among the edge proxies, namely *discovery*, *transcoding*, and *delivery* phases.

During the *discovery* phase, the edge servers cooperate to discover the version of the Web object requested by the client. Since many versions of the same object may exist in the proxies, it is necessary to carry out a multi-version lookup process that may require cooperation among the edge proxies. The discovery phase includes a local lookup and may include an external lookup. It works as following. Once the proxy server has determined the client capabilities, it looks for a copy of the requested resource in its cache. One of the following three events may occur. (1) *Local exact hit*: the proxy contains the exact version of the object that can be immediately delivered to the client. (2) *Local useful hit*: the proxy cache contains a more detailed and transcodable version of the requested object that can be transformed to obtain a less detailed version that meets the client request. Depending on the transcoding cooperation scheme, the proxy can decide either to perform the transcoding task locally or to activate an external lookup to look for an exact copy of the object. (3) *Local miss*: the proxy cache does not contain any valid copy of the requested object. This may be due to two cases: no version of the object is found, the existing version cannot be adapted to the client request. In both instances, the proxy must activate an external lookup that returns one of the following three results. (1) *Remote exact hit*: a remote proxy holds the exact version of the object, which is transferred to the requesting proxy during the successive delivery phase. (2) *Remote useful hit*: a remote proxy cache contains a more detailed and transcodable version of the object that can be transformed to meet the client request. Depending on the transcoding cooperation scheme, the edge server can decide either to perform the transcoding task locally or to provide the useful version to the requesting proxy. (3) *Remote miss*: no remote proxy contains any valid copy of the object. When a *global cache miss* occurs, the client request needs to be forwarded to the content Web server.

The *transcoding* phase is specific to the problem considered in this paper. When necessary, it includes the execution of some transcoding operations. Any proxy of the cooperative system is equipped with software to perform the transcoding operations required by any device type.

Once an exact version of the requested object is found (or generated), the *delivery* phase transfers the resource to the client. The final delivery is always carried out by the edge proxy that is first contacted by the client. Hence, if the object is found in another proxy, the delivery phase includes the object transmission to the edge proxy.

To have a general system that does not involve the content provider in the transcoding process, we assume that the Web server returns the original version of the object to the requesting proxy. This paper focuses on the discovery phase, where any proxy, that has to transcode a resource, stores in its cache both the retrieved and the transcoded versions of the object. We recognize that our architecture leaves open other possibilities that are worth of further investigation, such as cooperative pushing and replacement of objects.

We have implemented a prototype for the cooperative transcoding architecture that is based on Squid Web proxy cache, version 2.4. The technical details can be found in [2].

3 Cooperative discovery in a flat topology

In a flat topology, where all edge servers providing transcoding functionalities are *peers*, the resource discovery phase differs from traditional lookup because of two main peculiarities: multiple versions of the same object may exist in the caches; the multi-version lookup phase may return three results (i.e., cache miss, exact hit, and useful hit). While misses and exact hits are handled entirely during the cooperative discovery phase, a useful hit may activate some cooperative transcoding protocol. Space limits do not allow us to consider some interesting variants. Hence, we assume that when an edge proxy finds a useful version in its cache, it transcodes the object without activating any remote lookup process to search for an exact copy of the object.

Cooperative resource discovery has been studied for a while and many mechanisms have been proposed to address the related issues [9, 10]. Most of those mechanisms can be adapted to the lookup of multiple versions. The two main and opposite approaches for disseminating state information are well defined in the literature on distributed systems: *query-based protocols* in which exchanges of state information occur only in response to an explicit request by a peer; *directory-based protocols* in which state information is exchanged among the peers in a periodic way or at the occurrence of a significant event, with many possible variants in between. Extensions of a basic query-based protocol and a basic summary-based protocol (a simplified version of the directory-based protocols) are proposed in Sections 3.1 and 3.2, respectively.

3.1 Query-based discovery protocol

Query-based protocols are conceptually simple. When a cache server experiences a local miss, it sends a query message to all the peers in order 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 on the basis of which we implemented our multi-version discovery mechanism [2].

3.2 Summary-based discovery protocol

Directory-based protocols include a large class of alternatives. The two most important are the presence of one centralized directory vs multiple directories disseminated over the peers, and the frequency for communicating a local change to the directory/ies. We consider distributed directory-based schemes because it is a common view that in a geographically distributed system any centralized solution could easily become the bottleneck [9].

In a distributed directory-based scheme, 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. Distributing the directory among the cooperating peers avoids the polling of multiple proxies during the discovery phase and, in the ideal case, makes object lookup extremely efficient. However, the ideal case is affected by large traffic overheads for keeping the directories up-to-date. Hence, real implementations use multiple relaxations, such as compressed directories (namely, *summary*) and less frequent information exchanges for saving memory space and network bandwidth. We chose Cache Digests as a basis of our multi-version summary-based cooperation scheme, because of its popularity and implementation in the Squid software. The support for multi-version lookup in our prototype did not require significant modifications of the basic summary-based protocol. All details can be found in [2]. The basic mechanism of Cache Digests cooperation is preserved, although the lookup process becomes more expensive because a lookup for every possible useful version must be performed.

4. Experimental results

In this paper, we consider a classification of the client devices in six categories on the basis of their capabilities of displaying different objects and connecting to the assigned edge proxy [3]. The classes of devices range from high-end workstations/PCs which can consume every object in its original form, to midrange PCs or laptops, set-top boxes, hand-held PCs, personal digital assistants, and cellular phones. We consider that transcoding is applied only to image objects (both GIF and JPEG formats), as more than 70% of the files requested in the Web still belongs to this class (e.g., [4]). In the system model, we assume that traditional devices, such as PCs and laptops, are still more popular than other client devices; hence, 25% of client requests come from PCs not requiring any transcoding, 25% from laptops with low or null necessity of content adaptation, while the remaining 50% of requests is equally divided among the other four classes of devices with major transcoding requirements.

For the experiments, we use two workload models, namely *Workload 1* and *Workload 2*. Workload 1 is based on proxy traces belonging to the nodes of the IRCache infrastructure. We can assume that Workload 1 captures a realistic scenario, because we found that the statistical characteristics of the images contained in the workload (e.g., file size, JPEG quality factor, number of colors of GIF images) are very close to the results reported in [4]. Workload 2 is char-

acterized by image files much larger than those contained in Workload 1, thus requiring in average a transcoding cost about three times higher. Workload 2 does not aim at being realistic, but at verifying the performance results when the computational process for transcoding has a major impact on the cooperative system. From the file list of the workload model, we obtained 80 different traces that were used in parallel during the experiments. Each trace consists of 1000 requests with a random delay that elapses between two consecutive requests. We use a uniform distribution of the traces, so that each proxy receives the same number of requests. We also consider only complete transcoding relation graphs, where each version can be created starting from any higher quality version [3].

In our experiments we set up a system of 16 proxy servers that are equipped with our Squid-based prototypes, and are connected through a fast Ethernet switched LAN. The clients are well connected to the edge proxies, while the content servers are placed in a remote location. As the path to the content servers is a possible system bottleneck, the global hit rate may impact the response time.

We consider two flat peer-to-peer architectures where resource discovery is based on a modified version of the query-based protocol (**ICP**) and the summary-based protocol (**Cache Digests**). For performance comparison, we consider also the non-cooperative scheme (**No_Coop**), in which the edge servers do not cooperate for resource lookup and transcoding. In No_Coop scheme, exact and useful hits can only be local, while any local miss activates a request of the resource to the content Web server.

Figure 1 shows the cumulative distribution of the system response time, that corresponds to the interval between the instant in which the client sends a request to the edge server and the instant in which the client receives all the response. We can see that the query-based cooperation clearly reduces the response time with respect to the summary-based cooperation. For example, 80% of requests experiment a response time less or equal to 550 msec when ICP is used, while the 80-percentile for Cache Digests is about 1450 msec and it is beyond 2500 msec for No_Coop.

From related analyses, we can say that these performance differences are largely related to the different cache hit rates achieved by the discovery protocols. Table 1 shows that when cooperation occurs among many nodes, Cache Digests tends to be imprecise: the accuracy of the exchanged cache digests decreases and its remote hit rates diminish. For example, columns 4 and 5 in Table 1 show that the reduction in the global hit rate for Cache Digests is caused by remote hit rates lower than those of ICP.

In a second set of experiments we compared the performance of the cooperation architectures when they are subject to the computationally heavier Workload 2. As expected, we found that the presence of larger images in-

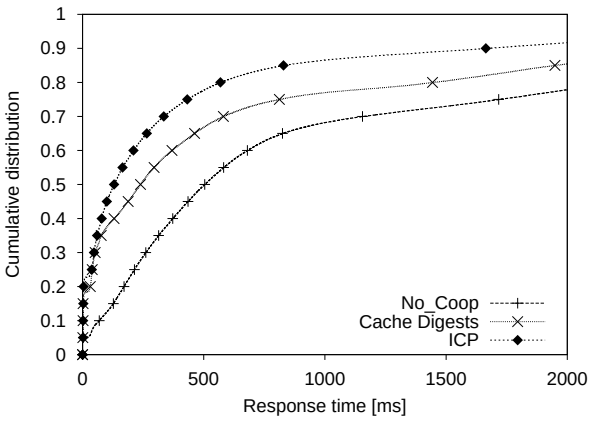

Figure 1. System response time (Workload 1).

	Local exact	Local useful	Remote exact	Remote useful	Global
No_Coop	5.3%	5.4%	n/a	n/a	10.7%
Cache Digests	5.5%	6.2%	13.9%	15.3%	40.9%
ICP	5.4%	6.0%	18.3%	28.7%	58.4%

Table 1. Cache hit rates (Workload 1).

creases the response time with respect to the curves shown in Figure 1, but it also showed an interesting and initially unexpected result. Although the global hit rate of a query-based protocol such as ICP decreases only slightly (from 58.4% to 52%), and it remains well above that of Cache Digests (from 40.9% to 29.7%), the response time of the ICP-based architecture tends to augment faster than that of Cache Digests. In particular, the cross-over of the two curves occurs around the 75-percentile. There are two reasons for this result. The cooperation cost of ICP is much higher than that of a summary-based protocol. When the system is heavily loaded, the higher CPU utilization of the proxies has a visible impact on the response time. Moreover, the number of queries that result in ICP timeout (2 seconds) increases above 10%.

We give a further confirmation of the previous conclusions by focusing on the response time limited to local and remote cache hits (exact and useful) for Workload 1 and 2. As we exclude the response times due to global misses, we can use a more controlled testing environment, which is not subject to external network variations, and put Web servers on the same LAN of proxies.

Figures 2 and 3 show the cumulative distributions of the hit response times for Workloads 1 and 2, respectively. As expected, the curve corresponding to the local hits overlap. Now, in both scenarios, Cache Digests obtains lower response times for remote hits than those of ICP. The higher cache hit rate of ICP does not compensate the overhead due to the processing of a larger number of messages that tend

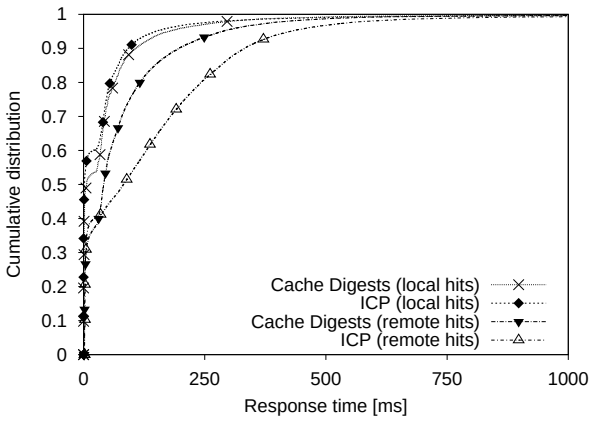

Figure 2. System response time for local and remote hits (Workload 1).

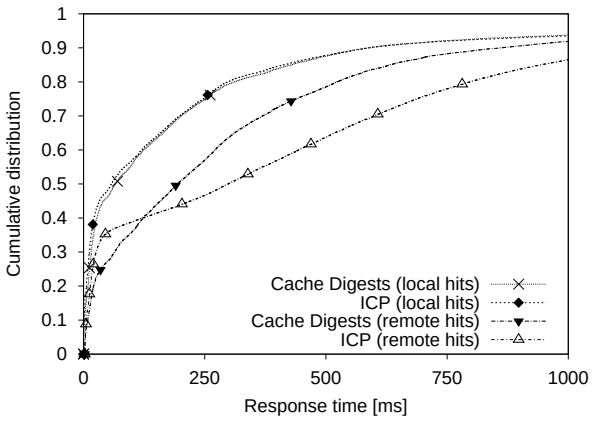

Figure 3. System response time for local and remote hits (Workload 2).

to overload the CPU, especially when the transcoding cost is high. This result is opposed to that obtained by experiments where ICP and Cache Digests are used in a context of traditional caching without transcoding. In that case, the CPU of the proxies is not subject to heavy load, hence the higher cache hit rate of ICP could easily compensate the higher overheads of query-based cooperation.

5. Conclusions

In this paper, we have investigated practical schemes for cooperative proxy caching and transcoding that can be implemented in the existing Web infrastructure. We have compared their performance through real prototypes based on Squid. A system consisting of edge proxies that cooperate in multi-version content caching, discovery, and transcoding opens up many interesting research issues. A limited number of problems have been investigated in this work, that to the best of our knowledge represents the first implementation of cooperative transcoding and caching systems for a flat peer-to-peer topology working with query- and summary-based protocols. We have found that cooperative schemes can substantially reduce the system response time compared to an architecture of non-cooperative proxies. Moreover, we have found that a query-based protocol such as ICP generally performs better with higher hit rates. On the other hand, a summary-based protocol such as Cache Digests, can provide better performance for heavier workload scenarios, because it experiences a lower cooperation overhead not requiring a large exchange of messages.

Acknowledgements

The first three authors acknowledge the support of MIUR-Cofin 2001 “High-quality Web systems” and MIUR-FIRB “Wide-scale, Broadband, Middleware for Network Distributed Services”.

References

[1] M. Butler, F. Giannetti, R. Gimson, and T. Wiley. Device independence and the Web. *IEEE Internet Computing*, 6(5):81–86, Sept./Oct. 2002.

[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. `http://weblab.ing.unimo.it/research/trans_caching.shtml`.

[3] V. Cardellini, P. S. Yu, and Y. W. Huang. Collaborative proxy system for distributed Web content transcoding. In *Proc. of 9th ACM Conf. on Information and Knowledge Management*, pages 520–527, 2000.

[4] S. Chandra, A. Gehani, C. S. Ellis, and A. Vahdat. Transcoding characteristics of Web images. In *Proc. of Multimedia Computing and Networking Conf.*, Jan. 2001.

[5] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-based scalable network services. In *Proc. of 16th ACM Symp. on Operating Systems Principles*, pages 78–91, Oct. 1997.

[6] S. Ihde, P. P. Maglio, J. Meyer, and R. Barrett. Intermediary-based transcoding framework. *IBM System Journal*, 40(1):179–192, 2001.

[7] 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.

[8] A. Maheshwari, A. Sharma, K. Ramamritham, and P. Shenoy. TransSquid: Transcoding and caching proxy for heterogeneous e-commerce environments. In *Proc. of 12th IEEE Workshop on Research Issues in Data Eng.*, Feb. 2002.

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

[10] D. Wessels. *Web Caching*. O’Reilly, 2001.