Distributed Cooperation Schemes for Document Lookup in Multiple Cache Servers*

Riccardo Lancellotti
Dip. di Informatica, Sistemi e Produzione
Università di Roma "Tor Vergata"

Bruno Ciciani
Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113, 00149 Roma
Tel. +39-06-49918325, Fax +39-06-85300849
ciciani@dis.uniroma1.it

Michele Colajanni
Dip. di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia

Università di Roma "Tor Vergata"
Dept. of Computer Engineering
TR-2002-22

December 2002

Abstract

Architectures consisting of multiple cache servers are a popular solution to deal with performance and network resource utilization issues related to the growth of the Web request. Cache cooperation is often carried out through purely hierarchical and flat schemes that suffer from scalability problems when the number of servers increases.

We propose, implement and compare the performance of three novel *distributed* cooperation models based on a two-tier organization of the cache servers. The experimental results show that the proposed architectures are effective in supporting cooperative document lookup and downloading. They guarantee cache hit rates comparable to those of the most performing protocols with a significatively reduced cooperation overhead. Moreover, in the case of congested network, they reduce the 90-percentile of the system response time up to nearly 30% with respect to the best pure cooperation mechanisms.

1 Introduction

Web caching has evolved as the first way to address Web server and network utilization issues related to the growth of HTTP requests. The basic idea of using one proxy server does not work well because of very low cache hit rates. Better performance can be achieved through interactions among various proxies. *Global caching* or *cooperative caching* architectures are used by public organizations (e.g., IRCache [8]), Internet Service Providers (e.g., AT&T [2]), third party companies, such as Content Delivery Networks (e.g., Akamai [1], Digital Island [4]). For some recent surveys, see [18, 11] or [10, 19].

The two most popular approaches for cooperative lookup refer to a hierarchy of cooperating caches (*hierarchical architecture*) or to a flat cooperation topology (*distributed architectures*). In hierarchical architectures a cache miss will result in looking for the resource to an upper level cache [22]. In distributed architectures every cache is supposed to be at the same level, and missed resources at one proxy are looked for in all cooperating cache servers. *Hybrid architectures* have been studied as well, for example in [11] and [14]. This latter work proposes and evaluates through a trace-driven simulator a new distributed architecture based on two-tier cooperation. The idea is to split the cooperative lookup process in two less expensive processes that involve subsets of the whole group of cooperating caches and different protocols.

In this paper, we give various contributions.

- We present and extensively test a prototype based on Squid [20] that implements a two-tier cooperation scheme similar to that described in [14].
- We also evidence the main differences among the simulation results and the experimental results obtained here. This different behavior is important because it motivates the search for novel cooperative architectures based on hybrid schemes.
- We describe the model and implementation of two novel two-tier schemes that address the issues raised from the experimental results of the first prototype.
- We evaluate the performance of the three prototypes and compare it against that of other protocols available in Squid, such as ICP [21] and Cache Digest [12].

The rest of this paper is organized as following. Section 2 describes the basic two-tier architecture and its implementation. Section 3 discusses some limitations of the basic approach and proposes two alternative hybrid schemes for cooperation.

The experiments described in Section 4 aim to verify that the novel protocols are a viable solution for cooperative Web caching. In particular, they offer high hit rates and limited network overhead even when the cooperation involves many nodes. This demonstrates that both novel architectures address in an effective way the scalability issues related to cooperative Web caching. Moreover they are a viable solution to cooperation in a heterogeneous environment, such as that of a wide area network. In Section 5 we discuss related work in distributed strategies for cooperative caching.

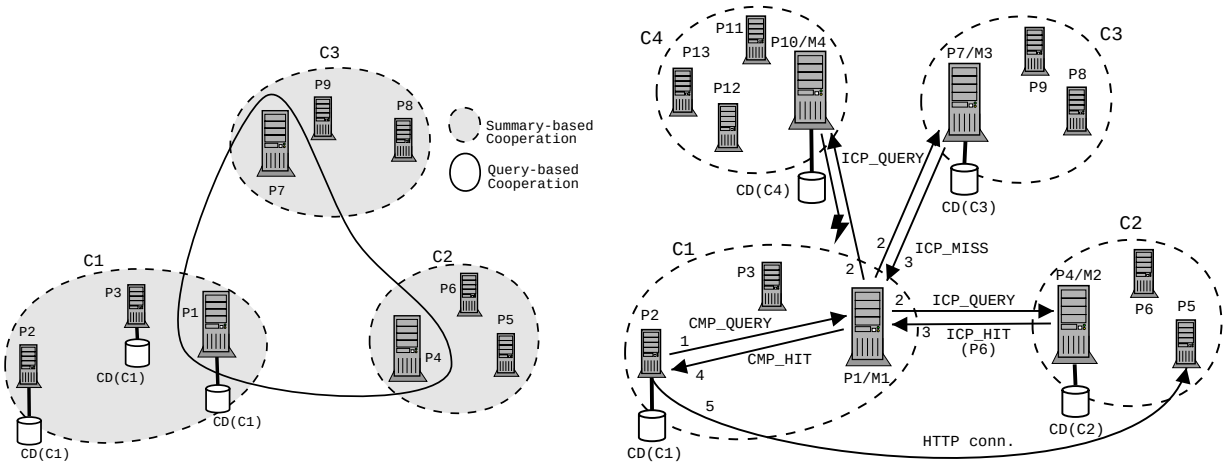
Finally, Section 6 presents some conclusions.

2 Two-tier Web caching architectures

Two-tier architectures are based on a partition of the cache servers into *clusters* that are mainly based on the structure and performance of the available network connections. As described in [14], two-tier cooperation combines informed-based and query-based protocols [10] to create a two-step, two-tier cooperative lookup. A client request reaching a cache server of a cooperative architecture using a two-tier organization of the servers may experiment different effects. Before proceeding further with the prototype description, we introduce some concepts related to the possible scenarios occurring in a two-tier lookup process. A client request can result in a *local hit* when a valid copy of the requested resource is found in the first contacted cache server: the document is sent to the client and no cooperation is activated. A *first-tier hit* occurs when the requested resource is found inside a subset of cache servers through the 1-st tier cooperation scheme, without activating the 2-nd tier lookup process.

A *global hit* occurs when the resource is found in a cache server by means of both cooperation tiers. Finally, we have a *global miss* if the resource has to be retrieved from the origin server, because both cooperation tiers fail in finding a valid copy of the resource in any cache server.

The first prototype implementing this cooperation mechanism is called Summary-Query because it uses a summary-based (1-st tier of cooperation) and a query-based protocol (2-nd



(a) Structure

(b) Operational example

Figure 1: Summary-Query architecture

tier) as the mechanism for intra-cluster and intra-cluster cooperation, respectively. A more detailed description of the various types of cooperation protocols is given in Section 5. There are some differences between the architecture described in [14] and the prototype proposed here. We use Squid 2.4 as a software basis and the most popular cooperation mechanisms implemented there. In particular, we use Cache Digests [12] (instead of Summary Cache as suggested in [14]) as the informed-based protocol and ICP [21] as the query-based protocol. In the Summary-Query scheme, for each cluster we select one server, called *master*, that typically is the node of the cluster offering superior computing power and better connections to other clusters. The master, as described in greater detail later on, acts as a gateway for second tier cooperation: each query message directed to other clusters must pass through the cluster master, as well as every resource coming from other clusters.

An example of the Summary-Query organization is described in Figure 1(a). We consider nine cache servers organized in three clusters, namely C_1, C_2 and C_3. Cluster C_1 consists of three nodes: P_1, P_2, and P_3, where P_1 is the cluster master. Other clusters are built in a similar way, that is $C_2 = \{P_4, P_5, P_6\}$ and $C_3 = \{P_7, P_8, P_9\}$. The three nodes $\{P_1, P_4, P_7\}$ compose the set of cluster masters in which the second tier cooperation is carried out through a query-based protocol.

Figure 1(b) shows how a document can be found through the Summary-Query mech-

anism. To show all possible scenarios, this figure includes a new cluster (C_4) which was missing in the architecture in Figure 1(a). Moreover, to better evidence the cluster masters, we added the labels M_1, M_2, M_3 and M_4 to the nodes P_1, P_4, P_7 and P_{10}, respectively. Let us assume that P_2 executed a summary-based lookup and that the requested resource has not been found neither in the local cache nor inside the cluster (*first-tier miss* case). The second-tier cooperation is then activated: a query, through the CMP protocol (a novel protocol similar to ICP but simpler and lighter, described later), is sent back to the cluster master M_1 (step 1) that, in its turn, sends ICP queries to all the other cache masters (step 2). Each of them executes a summary-based lookup to check whether the resource is in its cache or inside its cluster. The responses are then sent to the cache master M_1 through a slightly modified ICP message (step 3). Packet losses are detected by means of a time-out mechanism (as for the case of the response coming from M_4). If any of the responses reports a hit (e.g., from M_2), a CMP_HIT message containing the address of the cache server P_5 is sent back to the cache server P_2 (step 4). P_2 retrieves the requested resource from P_5 through an HTTP connection (step 5), and forwards it to the client.

To implement the Summary-Query cooperation protocol the main modifications to Squid are localized in the modules called *peer selection* and *neighbors* [20]. The *peer selection* module contains the routines to select a cache server that may hold the requested resource. It is activated when a client request results in a local miss or in a local hit with a stale file. This phase is called *cooperation* in Figure 2.

The peer selection module uses other modules that implement some cooperation protocols, such as ICP and Cache-to-Master (CMP). The latter is a new protocol that has been created to support the Summary-Query scheme. It is based on ICP, even though it is lighter and simpler than the original ICP protocol, and, unlike ICP, it is able to report a hit in a third cache server not involved in the actual CMP message exchange. In our example the CMP response flows from M_1 to P_2, but the protocol has to indicate a hit occurring on P_5: without proper extensions, ICP can only indicate a hit or a miss in M_1.

The *neighbors* module contains the data-structure definitions and the routines to manage the database of cooperating proxies and the statistics about their state. A CMP hit is referred to a cache that does not belong to the original cluster and that is unknown to the cache that received the user request. The default Squid behavior is to reject hits from

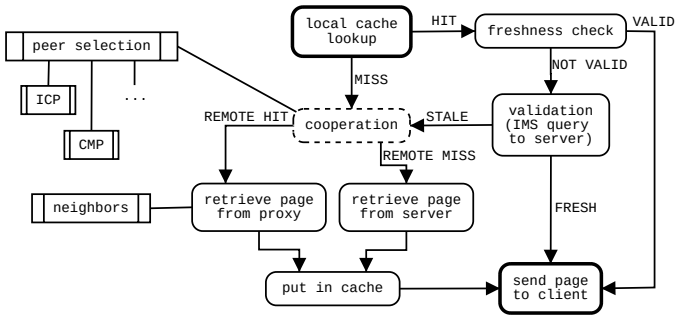


Figure 2: Service of a client request by Squid.

unknown caches for security reasons. Instead, a CMP_HIT message coming from the cluster master must override this behavior. As a consequence, to allow the resource retrieval even when such a message is received, the unknown peer is dynamically added to the list of known caches.

Figure 2 shows how Squid works when servicing a request from a client: a client request resulting in a local miss or in a local hit with a stale file forces the proxy server to retrieve the resource. Before contacting the origin server, some cooperative operations are activated in the *cooperation* phase described in Figure 2.

3 Novel schemes for hybrid cooperation

When we evaluated the performance of the Summary-Query scheme through a real prototype, we found that it can achieve high cache hit rates, but it places a significant amount of work on the cluster masters. Each master has to serve client requests and has to work as a gateway for query-based cooperation. Especially when cluster masters do not have computational capacity higher than that of the other cache servers, we found that this twofold role can easily congest these nodes to the extent that the entire cooperation system tends to have high response time. It is worth to observe that this risk was not evidenced by the simulation results reported in [14], but only by an experimental evaluation of which we outline the results in Section 4.2. Network researchers using simulation as a main evaluation tool should be aware of the risks of not considering the computational cost of network-based operations at the server side. For example, a misuse of the popular *ns-2* simulator [16] not including realistic server delays may lead to similar mistakes.

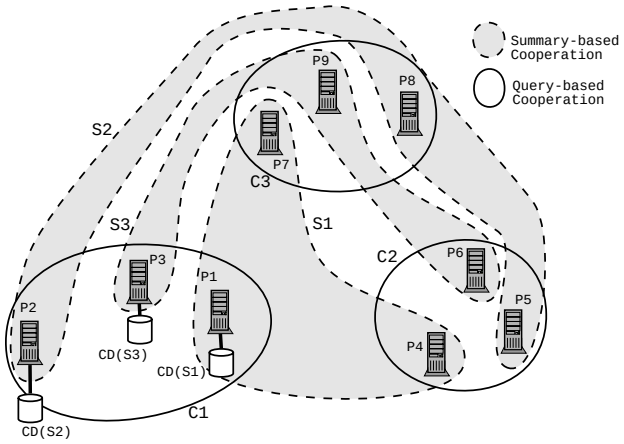


Figure 3: Query-Summary architecture

The rather poor performance results achieved by the Summary-Query scheme motivated the search for novel architectures based on hybrid cooperation that avoid congestion of the cluster master node. They are described in the following subsections.

3.1 SummaryMaster-Query cooperation

The risk of bottleneck is due to the twofold role of the cluster master. Hence, the simplest solution is to use *dedicated masters*, even though the price is a reduced number of cache servers available for caching operations. This approach denotes a different cooperation architecture, called *SummaryMaster-Query*, that works similarly to the previous Summary-Query scheme. The main difference is that now each master cache acts as a gateway for its cluster and as a directory for the other clusters, but it cannot be directly contacted by clients. It is interesting to note that this scheme can be considered as a representative of informed-based architectures that use one or multiple directories managed by dedicated servers.

The prototype of the SummaryMaster-Query scheme is fully interoperable with the Summary-Query scheme. Hence, it is easy to think to a mixed scheme, where some cache masters are dedicated, while others can accept client requests, depending on the expected level of congestion.

3.2 Query-Summary cooperation

Query-Summary is a quite different alternative to avoid poor performance of Summary-Query. It activates first-tier cooperation among clusters through a summary-based protocol and, if necessary, second-tier cooperation in a cluster through a query-based protocol. The motivation for Query-Summary with respect to Summary-Query is twofold: achieving better load balancing by eliminating cluster masters and driving the large majority of traffic to the best performing links. Indeed, Query-Summary scheme uses multiple representatives of each cluster for the requests to avoid the risk of poor performance caused by overloaded cluster masters. Moreover, the need of a fairer network resource utilization arises from the observation that in the Summary-Query schemes the large majority of traffic for cooperation (due to ICP messages) transits through the inter-cluster links, that are potentially slower and more expensive (for the ISPs) than intra-cluster links. Hence, in *Query-Summary*, inter-cluster cooperation is carried out through the Cache Digests protocol that generates a minor traffic, while communications inside a cluster, where nodes are connected through faster and less expensive links, are based on (enhanced) ICP messages.

Figure 3 shows an example of this cooperation scheme by considering the same architecture shown in Figure 1(a). The three clusters are $C_1 = \{P_1, P_2, P_3\}$, $C_2 = \{P_4, P_5\, P_6\}$, and $C_3 = \{P_7, P_8, P_9\}$. We also define the following *sets* for query-based cooperation: $S_1 = \{P_1, P_4, P_7\}$, $S_2 = \{P_2, P_5, P_8\}$ and $S_3 = \{P_3, P_6, P_9\}$. Each set must contain at least one member of each cluster. In this version of Query-Summary cooperation, we require that clusters C_i and sets S_i denote a *partition* of the initial set of nodes. This leads to the conclusion that the maximum number of sets is equal to the minimum cardinality of the clusters. Additionally, the minimum number of nodes inside a set S_i is equal to the number of clusters in the system, because at least a member of each cluster must belong to a set.

A local miss in a cache server (for example, P_1) triggers the lookup process on the digests that can lead to a hit inside the set S_1 to which P_1 belongs (the hit could be on P_4 or P_7). Alternatively, in the case of a first-tier miss, the cooperation is activated inside the cluster C_1. A query is sent to every member of the cluster that is, to P_2 and P_3 in the example of Figure 3. The cache server receiving the ICP request can reply with a hit, a miss or a pointer message when a digest lookup locates a hit inside its set. In our prototype, this last message is sent through the flag ICP_MISS_POINTER, as documented in [19]. In our example, P_2

can reply with a hit, if it owns the requested resource or with a pointer to P_5 or P_8, if it finds a hit on those nodes, otherwise it sends a miss message. Similar behavior is expected from P_3.

4 Experimental Results

All prototypes were extensively tested to verify their scalability and to compare their performance with that of the pure query-based and summary-based protocols. We first compared the various Two-tier architectures among each other to find out the best performing one. Then, we compard it with other distributed cooperative architectures. We settled two experimental testbeds: one aims to measure the response times of client requests, hence the servers are distributed over a geographic scenario; the second focuses on cache hit rates and cooperation overheads, hence we can use cache servers placed on the same network segment.

4.1 Workload models

The difficulty of defining a "typical" workload model is a well known issue, because studies on real traces show great differences. For the experiments, we prefer to use two synthetic workload models generated by Web-Polygraph version 2.5 [13]. They intend to capture two "realistic" Internet scenarios, that are based on the model proposed for the second cache-off by IRCACHE [8].

Workload 1 represents a set of heterogeneous users with different interests. This characteristic, together with the document popularity model, leads to a high spatial locality. Client requests also show some temporal locality, so that only 30% of the workload is active at a given time. Requests are referred to a mix of content types consisting of images (65%), HTML documents (15%), binary data (0.5%), others (19.5%). The workload model defines a set of *hot* resources (1% of the working set) that receive about 10% of the requests. The heterogeneity of the user interests is represented by the fact that only 50% of the requests is taken from a "public" set of pages common to all clients, while the remaining 50% is taken from a "private" set, different for each client. Each client is configured to visit more than once only 80% of the URLs and the object cacheability is 80%. HTML resources typically contain embedded objects.

Workload 2 is a modified version of Workload 1, where the client population is more homogeneous. Here, clients share the same interests, so spatial locality is reduced, while the overall access locality is augmented. Moreover, the popularity model is chosen so to increase the size of the hot fraction of the workload. This workload takes less advantage from local cache hit rates and puts more pressure on cooperation. The modified main parameters of Workload 1 are indicated below. The hot set of the workload is larger (15% of the access are referred to a hot set corresponding to 5% of the URL-space). This creates a popularity distribution with a heavier tail. The working set size keeps growing trough the whole experiment (there is no temporal locality). The fraction of public requests is 100%, which means that every document in the trace may be requested by any client.

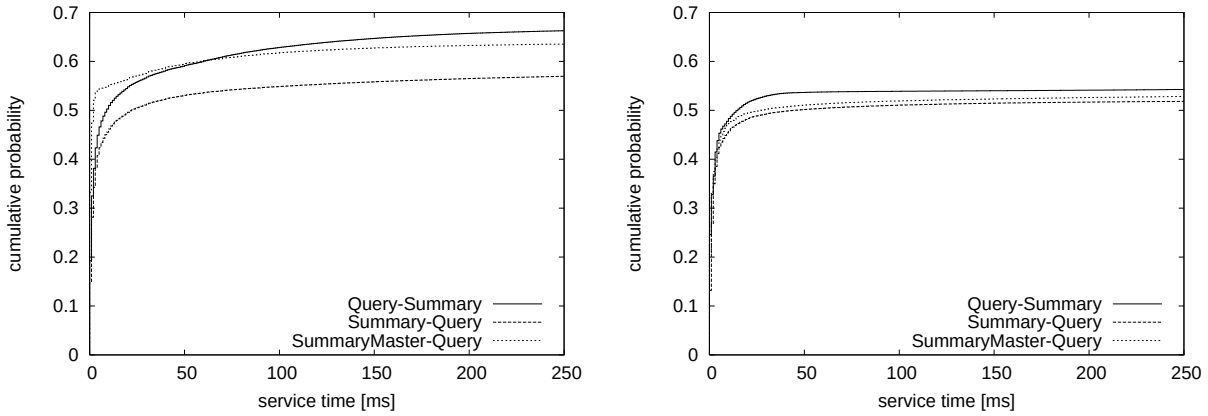
Every experiment was done twice and data measures were collected only in the second run, so to emulate a steady state scenario. It is worth to observe that an increment of the number of cache servers increases the global cache capacity, but also the size of the working set, because the number of clients is incremented proportionally, so that five clients are always connected to each cache server. Preliminary tests were done to tune some parameters of Squid. For example, for summary-based cooperation we find more convenient to use a digest rebuild period of 60 seconds to reduce the consistency miss effects that are due to frequent cache object replacements (note that the Squid default value is 60 minutes).

4.2 Comparison of Two-tier architectures

With the first set of experiments we aimed at comparing the performance of different two-tier architectures in terms of cache hit rate, cooperation overhead, and response time at the client.

4.2.1 User response time

The first set of experiments refers to a geographic testbed consisting of two clusters of five nodes each, connected through some links and a geographical backbone. Ten network hops existed between the clusters. We also installed one Web server in each location. For these experiments, we used Workload 1 and the response time as the performance metric. The experiments were carried out for two network conditions: the first test was executed on Sunday, with little other network traffic, while the second test was done in conditions of



(a) Light network traffic

(b) Heavy network traffic

Figure 4: System Response Time

heavy network load during a working day. The observed mean round-trip time was 50 ms and 300 ms, in the case of *light* and *heavy traffic*, respectively. We also monitored the network resource usage on the worst performing links (one link of the geographic interconnection has a capacity of 2 Mbit/s).

Figures 4 show the cumulative probability of the response times of all requests. In particular, Figure 4(a) and 4(b) refer to the light and heavy network traffic, respectively. From Figure 4(a), it is interesting to note the hints of congestion in the Summary-Query response time: the percentage of requests served within 1 second is 69% for Query-Summary, 65% for SummaryMaster-Query but only 59% for Summary-Query. This bad performance is due to the cluster masters that become the system bottlenecks, as it typically occurs for centralized components of distributed systems. Both alternative architectures proposed in this paper can effectively avoid this problem. Query-Summary achieves lower response times than those of Summary-Query, even if its object hit rate tends to be the worst, as shown in next section. As expected, in the case of heavier traffic, the overall performance is reduced. However, there are some noteworthy behaviors: the general link congestion reduces the performance difference among the various protocols. Moreover, looking at the 90-percentile of the response time, we can see that Query-Summary performance are worse than that of SummaryMaster-Query (5.6 vs. 4.4 seconds). This can be explained by considering that in Query-Summary even the 1-st tier lookup process tends to find hits in other physical clusters.

Table 1: Cache hit rates and overheads for Workload 2

Number of Nodes	Local HR	First-tier HR	Global HR	Intra-Clust Overhead per request [$\frac{bytes}{req.}$]	Inter-Clust Overhead per request [$\frac{bytes}{req.}$]	Total Overhead per request [$\frac{bytes}{req.}$]	Relative Overhead per node [$\frac{bytes}{req.}$]
Query-Summary							
8	38.38%	45.69%	60.16%	284.25	7.32	291.58	36.44
15	25.89%	37.87%	55.66%	427.38	18.10	445.48	29.70
30	15.13%	30.42%	51.04%	543.05	49.10	642.14	21.40
SummaryMaster-Query							
8	41.00%	52.50%	60.13%	62.05	41.67	104.72	13.09
15	23.63%	36.16%	60.96%	83.40	118.87	202.36	13.49
30	13.49%	26.71%	60.26%	116.61	249.58	366.19	12.21

Hence, the retrieval process is potentially more expensive, especially when the network is overloaded.

4.2.2 Cache hit rates (HR) and overheads

The second set of experiments aims at comparing hit rate and coordination overhead of the two-tier protocols. We choose not to perform our experiments on Summary-Query because it is very similar to SummaryMaster-Query and because from the previous section we found that it does not achieve acceptable user response times.

Table 1 shows the results of cache hit rate and cooperation overhead for Workload 2, while Table 2 shows the same results for Workload 1. In each table, the first three columns report the cache hit rates (local, first-tier and global), while the last columns report the traffic overhead due to cooperation, that is indicated as additional bytes exchanged for each client request.

Actually, the most interesting results for evaluating the cooperation schemes are obtained for Workload 2. We derive the main conclusions from these experiments and use the results obtained from Workload 1 to confirm or show some light differences. We have already observed that increasing the number of cache servers leads to an increment of the working set size and a consequent reduction of the percentage of documents that can be cached in each node. For example, for Workload 2 these percentages decrease from 7.5% to 2% of the working set when the cooperating nodes pass from 8 to 30. However, the most interesting

Table 2: Cache hit rates (HR) and overheads for Workload 1

Number of Nodes	Local HR	First-tier HR	Global HR	Intra-Clust Overhead per request $[\frac{\text{bytes}}{\text{req.}}]$	Inter-Clust Overhead per request $[\frac{\text{bytes}}{\text{req.}}]$	Total Overhead per request $[\frac{\text{bytes}}{\text{req.}}]$	Relative Overhead per node $[\frac{\text{bytes}}{\text{req.}}]$
Query-Summary							
8	55.01%	57.60%	63.84%	332.15	7.14	339.29	42.41
15	39.81%	43.82%	52.55%	317.66	10.88	328.54	21.90
30	35.80%	44.11%	54.38%	427.40	24.46	451.86	15.06
SummaryMaster-Query							
8	48.24%	55.40%	62.56%	70.44	82.84	153.28	19.16
15	40.77%	50.53%	63.62%	65.54	95.09	160.64	10.71
30	31.65%	55.12%	64.06%	88.37	192.83	281.20	9.37

aspect of the Workload 2 characteristics is that a larger number of peers leads to a sensible reduction of the local cache hit rates. From Table 1 we have that the local cache hit rates go from about 40% to about 15%. A similar reduction can also be found for Workload 1 (Table 1), but this effect is less evident because of higher document locality.

Both two-tier architectures can compensate the reduction of the local hit rate by means of the cooperation mechanism. However, the SummaryMaster-Query approach achieves higher scalability: as the number of peers grows, the first-tier hit rate drops and the second-tier can only partially compensate this decline (refer to columns 3 and 4 of Table 1 and 2). On the other hand, Query-Summary seems to offer a greater hit rate, especially for higher numbers of nodes.

The most interesting differences between the two cooperation approaches can be found in their capacity to distribute overheads due to cooperation. Both schemes show a good scalability because their overhead grows much slower than the number of nodes involved in cooperation (as the last column of Table 1 and 2 shows). However, Query-Summary overhead is almost the double of the SummaryMaster-Query overhead, as shown in column 7 of Table 1 and 2. It is worth to observe that inter-cluster overhead of Query-Summary is much lower because of effectiveness of the summary-based cooperation scheme in reducing the size of exchanged information. Hence, Query-Summary can be very useful in systems where inter-cluster network resources must be spared, for example when many peering points are involved.

4.3 Comparison with other cooperation protocols

In this section, we aim to verify whether hybrid cooperation protocols based on a two-tier topology can scale better than pure protocols. We start by illustrating the results of our experiments on hit rate and cooperation overhead and then we will show how those data are related with user requests service times. From the previous section we have found that SummaryMaster-Query can achieve good performance that is, low response time and high cache hit rate. Therefore, we use this cooperation mechanism as a representative for the whole class of two-tier architectures.

4.3.1 Cache hit rates and overheads

Table 3: Cache hit rates (HR) and overheads for Workload 2

Number of Nodes	**Local HR**	**Global HR**	**Total Overhead per request** $[\frac{\text{bytes}}{\text{req.}}]$	**Relative Overhead per node** $[\frac{\text{bytes}}{\text{req.}}]$
SummaryMaster-Query				
8	41.00%	60.13%	104.72	13.09
15	23.63%	60.96%	202.36	13.49
30	13.49%	60.26%	366.19	12.21
Cache Digest				
8	38.00%	53.05%	5.70	0.71
15	26.92%	44.29%	6.32	0.42
30	16.28%	33.77%	6.84	0.23
ICP				
8	40.54%	66.49%	780.84	97.60
15	27.07%	66.15%	1882.64	125.51
30	16.19%	65.20%	4500.42	150.14
No Cooperation				
8	49.64%	49.64%	n/a	n/a
15	27.84%	27.84%		
30	16.35%	16.35%		

To evaluate the scalability of the two-tier protocols, we compare cache hit rates and cooperation overheads for an architecture with an increasing number of cache servers that are subject to Workload 1 and Workload 2. As a comparison testbed, we also include results for an architecture with the same number of nodes that do not cooperate for document lookup (namely, No Cooperation). Table 3 and Table 4 summarize the results with respect

Table 4: Cache hit rates (HR) and overheads for Workload 1

Number of Nodes	**Local HR**	**Global HR**	**Total Overhead per request** $[\frac{\text{bytes}}{\text{req.}}]$	**Relative Overhead per node** $[\frac{\text{bytes}}{\text{req.}}]$
SummaryMaster-Query				
8	48.24%	62.56%	153.28	19.16
15	40.77%	63.62%	160.64	10.71
30	31.65%	64.06%	281.20	9.37
Cache Digest				
8	31.47%	40.25%	3.67	0.46
15	30.52%	37.82%	5.11	0.34
30	29.14%	37.80%	5.48	0.18
ICP				
8	57.22%	69.46%	532.72	66.59
15	52.16%	68.24%	1238.13	82.54
30	45.12%	67.62%	2977.00	99.23
No Cooperation				
8	57.30%	57.30%		
15	52.28%	52.28%	n/a	n/a
30	45.45%	45.45%		

to Workload 2 and 1, respectively. The structure of those tables is similar to that of Table 1 and Table 2, even if values referring to the two-tier scheme have been removed). The first two columns report the cache hit rates, while the last columns report the traffic overhead due to cooperation, that is indicated as additional bytes exchanged for each client request.

The reduction in local hit rate for higher numbers of nodes occurs as in previous experiments. Some differences dependent on the cooperation scheme were expected, because a cache server belonging to a cooperative system receives requests from its clients as well as from other cache servers. This alters the access patterns and in practice reduces the document locality. Indeed, the best local hit rates are obtained by the No-Cooperation scheme that preserves locality at most. Same behavior is shown in Table 4 referring to Workload 1, although in this case the local hit rates remain higher than those observed for Workload 2. The larger space of improvement for cooperation motivates our main interest for this latter model.

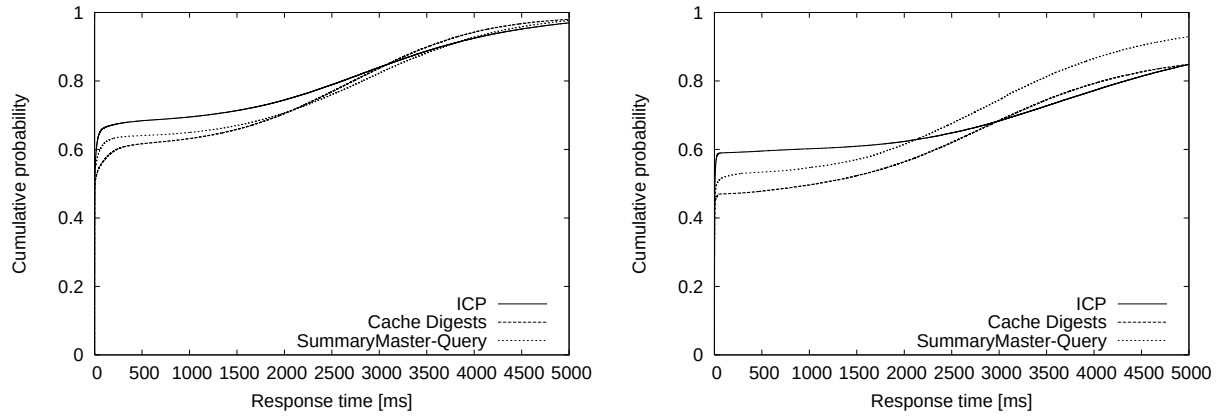
From Table 3 we can observe that the No-Cooperation scheme is not able to face the reduction of the local cache hit rates, and also the Cache Digests scheme shows many limits,

especially for higher numbers of peers. On the other hand, ICP and the hybrid cooperation scheme compensate the effects of the local hit rate reduction with a global cache hit rate that is stable and always over 50%. The cache hit rate of ICP is slightly higher (up to 5%) than that of hybrid schemes, but the price is an excessive cooperation overhead, as shown in column 4 of Table 3 and 4. Moreover, the motivation for the better hit rate of ICP is also related to the choice of the workload models that tend to penalize summary-based cooperation. Indeed, an increment of the working set size rises the frequency of object replacement, thus reducing the accuracy of the exchanged Cache Digests (there is an increment of capacity and consistency misses). Cache Digests is particularly sensitive to this and its hit rate (quite low even with only 8 cooperating nodes) is further reduced when the working set size increases. The performance degradation of summary based-cooperation was another unexpected result that the simulator described in [14] was unable to show. ICP is not affected by the frequency of cache replacements and this explains its better hit rates. It is remarkable how well the hybrid schemes are able to address the issues related to cache replacement, even if the peculiarities of the workload model affect their summary cooperation component occurring on the first-tier.

In results related to Workload 1, because of the higher spatial locality in request patterns, the hit rates are generally higher than those shown in Table 3. The main contribution to these results are due to the local hit rate (see No-Cooperation performance), which is increased by the greater locality caused by the higher percentage of "private" requests, as explained in the description of the workload models. Cooperation tends to reduce locality, and this motivates the results of Cache Digest that are even poorer than those of No-Cooperation.

If we include the overheads due to cooperation in the analysis of scalability, we can observe some interesting modifications in the position of the cooperation schemes. Once again, we use Workload 2 (Table 3) as a main reference to explain our conclusions, and report data for Workload 1 (Table 4) as a comparison testbed.

As expected, augmenting the number of cache servers increases the cooperation overhead, but not every scheme is affected in the same way. In particular, ICP generates the highest cooperation traffic even with 8 nodes, and continues to increase it with respect to the number of peers in a much faster way than any other scheme does. It is interesting to note that the ICP cooperation overhead (traffic generated only to cooperative lookup purposes) for 30



(a) Light network traffic

(b) Heavy network traffic

Figure 5: System Response Time

nodes reaches 4.5 KByte per requests. With an average document size of about 10 KByte, this means an increment of almost 50%. On the other hand, Cache Digests shows the lowest cooperation overhead.

Hybrid schemes occupy an intermediate position, with a cooperation overhead higher than that of Cache Digest, but significantly lower than that of ICP. The results observed for Workload 2 are substantially confirmed by the experiments based on Workload 1, as shown in the last columns of Table 4. For appreciating the scalability of the cooperation schemes, we can also refer to the last column of Tables 3 and 4, showing the overhead traffic per request relative to each node. It is important that for larger numbers of peers, the two-tier architectures and Cache Digests relative overheads tend to diminish, whereas it increments exponentially for ICP.

4.3.2 User response time

Experiments in the geographic scenario have also been carried out on traditional cooperation mechanism to compare their response time with that offered by hybrid cooperation protocols.

Indeed, there is a big difference between client requests resulting in a hit (served usually in very short time) and requests occurring in a global miss (that usually experience a much higher latency). Hence higher hit rate means that higher percentages of requests can be

served in a short time. ICP shows the best response times for hit documents, but poorest results in case of global misses, because it has to wait for the response from the slowest peer or the time-out expiry. In Figure 5(a) the ICP curve is the highest at the beginning, but it is surpassed by that of Query-Summary (around 1.5 seconds, not shown), by SummaryMaster-Query (after 2 seconds) and becomes the worst (after 4 seconds). These effects, even if already shown by our experiments, became much more evident in other not reported curves obtained for a workload model characterized by scarce locality and consequent lower hit rates. The conclusions on response times can be better appreciated by evaluating the 90-percentile of the request service time instead than looking at the curves that seem so close. In the case of light load, the 90-percentile for ICP is 3.7 seconds (the highest value), while for Query-Summary it is 3.2 seconds and for SummaryMaster-Query it is 3.6 seconds.

When heavy traffic is placed on the network (Figure 5(b)), congestion can occur in many places on the mesh. The overall performance of the cooperative system is reduced. In particular, the ratio of client requests serviced within 100 ms is reduced from 66%-56%, depending on the cooperation model used, to 59%-47%. Also the cache hit rate decreases from 68%-47% to 61%-36% when the network is heavily loaded. Clearly, the 90-percentile of the user response time augments: ICP shows once again poor performance (6.2 seconds), far higher than that of the best SummaryMaster-Query remaining below 4.5 seconds. Cache Digests is the worst cooperation approach in terms of 90-percentile (7.3 seconds) when the network is congested. The motivation is that its lower hit rate augments the use of congested links to contact the Web servers.

5 Related work

Cooperation in distributed architectures can occur in many different ways. However, most of the proposed solutions can be classified as *query-based or* informed-based cooperation mechanism.

In query-based protocols each client request result in a query message sent to every cooperating node (also called *peer*). The most important protocol belonging to this group is ICP, described in [21]. HTCP [17] is an evolution of ICP studied to add support for the complex caching semantic introduced with HTTP 1.1.

The main idea behind informed-based cooperation is that status information exchange

(usually the cache index) occurs *before* client requests, hence remote lookup can be done using locally stored information. This cooperation is more complex than query-based cooperation because more choices are available. In particular, two main questions should be answered when designing such protocols: when information are exchanged and how cache content is coded. Information exchange can be synchronous (when every significative change in status is notified to peers) or asynchronous (when data exchange occurs in a loosely periodic way). Moreover status information can be the whole cache index or a summary of it, often obtained using some sort of *lossy* compression such as Bloom filters [3]. The available possibility ranges then from synchronous cache index exchange to asynchronous summary-based information exchange. Cache Digests [12] is an important asynchronous summary-based cooperation protocol that is available in Squid. Another well-known cooperation protocol is Summary-Cache [5], that is similar to Cache Digests, but with a query mechanism used to check potential hits to reduce the risk of false hits. Another popular informed-based protocol is CRISP [7, 6], where a centralized directory is used to reduce the overhead related to less compressed status information being exchanged.

All mechanisms using one pure cooperation protocol suffer from the lack of scalability, especially in a geographic context, where some links can be congested or can offer very different types of bandwidth. The limitation of ICP is mainly related to the high traffic introduced for cooperation that grows quadratically as the number of peers increases [5]. Moreover, increasing the number of nodes augments the risk of packet loss or delay, hence misses are detected more slowly, as suggested in [12] and as our experimental results confirm.

In summary-based cooperation, we observed a clear trade-off between information accuracy and cooperation overhead. In general, we found that Cache Digests is very effective in reducing cooperation overhead, but the offered hit rate is quite low. On the other hand, CRISP based on a centralized directory, is not suitable for a geographic environment where a centralized information repository can become a bottleneck and a single point of failure, as noted by the authors themselves [9].

A completely different approach in cooperation is CARP [15] where hashing is used to partition the URL-space between the cache servers. However, this solution is not suitable for a geographic environment where network status changes dynamically in a significative way.

6 Conclusions

This paper offers multiple contributions. We investigate the benefits in cache hit rate and response time of a two-tier cooperation previously studied only by means of simulations. An under evaluation of CPU costs at the cache servers for network operations caused some differences between the previously obtained simulation results and the experimental results. (Other network-oriented simulations could be affected by similar problems.) In particular, the prototype shows congestion problems at some servers that are solved by the two novel two-tier mechanisms proposed in this paper. Our experiments show that the modified two-tier prototypes offer higher hit rate with respect to Cache Digests and lower cooperation overhead than ICP. The stability of the response time of two-tier architectures can be appreciated also in a geographic environment with heterogeneous links.

References

[1] Akamai. Akamai inc., 2002. `http://www.akamai.com`.

[2] ATT. At&t, 2002. `http://www.att.com`.

[3] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, Jul. 1970.

[4] DigitalIsland. Digital island inc., 2002. `http://www.digitalisland.com`.

[5] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[6] S. Gadde, J. Chase, and M. Rabinovich. A taste of crispy squid. In *Proc. of Workshop on Internet Server Performance (WISP'98)*, 1998.

[7] S. Gadde, M. Rabinovich, and J. Chase. An approach to building large internet caches. In *Proc. Sixth Workshop on Hot Topics in Operating Systems (HotOS-VI)*, May 1997.

[8] IRCache. Ircache project, 1995. `http://www.ircache.net`.

[9] M. Rabinovich, J. Chase, and S. Gadde. Not all hits are created equal: Cooperative proxy caching over a wide-area network. In *Proc. of Third International WWW Caching Workshop*, Jun. 1998.

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

[11] P. Rodriguez, C. Spanner, and E. Biersack. Web caching architectures: hierarchical and distributed caching. In *Proc. of Web Caching Workshop (WCW'99)*, 1999.

[12] A. Rousskov and D. Wessels. Cache digests. *Computer Networks and ISDN Systems*, 30(22-23), Nov. 1998.

[13] A. Russkov and D. Wessels. Web polygraph, 2000. `http://www.web-polygraph.org`.

[14] A. Santoro, B. Ciciani, M. Colajanni, and F. Quaglia. Two-tier cooperation: A scalable protocol for web cache sharing. In *Proc. of IEEE International Symposium on Network Computing and Applications*, Oct 2001.

[15] V. Valloppillil and K. Ross. Cache array routing protocol v1.0, Feb. 1998.

[16] P. VINT. Ns2, 2002. `http://www.isi.edu/nsnam/ns/`.

[17] P. Vixie and D. Wessels. Hyper text caching protocol (htcp/0.0), Jan. 2000. RFC 2756.

[18] J. Wang. A survey of web caching schemes for the internet. *ACM Computer Communication Review*, 29, Oct. 1999.

[19] D. Wessels. *Web Caching.* O'Reilly, 2001.

[20] D. Wessels. *Squid Programmers Guide*, 2002.

[21] D. Wessels and K. Claffy. Internet Cache Protocol (ICP), version 2, Sep. 1997. RFC 2186.

[22] P. S. Yu and E. A. MacNair. Performance study of a collaborative method for hierarchical caching in proxy servers. *Computer Networks and ISDN Systems*, pages 215–224, Apr. 1998.