Distributed Cooperation Schemes for Document Lookup in Multiple Cache Servers

Riccardo Lancellotti
Dipartimento di Informatica,
Sistemi e Produzione
Università di Roma "Tor Vergata"
riccardo@weblab.ing.unimo.it

Bruno Ciciani
Dipartimento di Informatica
e Sistemistica
Università di Roma "La Sapienza"
ciciani@dis.uniroma1.it

Michele Colajanni
Dipartimento di Ingegneria
dell'Informazione
Università di Modena e Reggio Emilia
colajanni@unimo.it

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 download. They guarantee cache hit rates comparable to those of the most performing protocols with a significant reduction of the cooperation overhead. Moreover, in 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 [19, 11] or [10, 20].

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 [23]. 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 [21] 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 [22] and Cache Digest [12].

The rest of this paper is organized as following. In Section 2 we discuss related work in distributed strategies for cooperative caching. Section 3 describes the basic two-tier architecture and its implementation. Section 4 discusses some limitations of the basic approach and proposes two alternative hybrid schemes for cooperation.

The experiments described in Section 5 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.

Finally, Section 6 presents some conclusions.

2. 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 mechanisms.

In query-based protocols each client request results in a query message sent to every cooperating node (also called *peer*). The most important protocol belonging to this group is ICP, described in [22]. HTCP [18] is an evolution of ICP thats support the complex caching semantic introduced by the HTTP/1.1 protocol.

The main idea behind informed-based cooperation is that the status information exchange (usually the cache index) occurs *before* client requests, hence remote lookup can be based on locally stored information. This cooperation is more complex than that of query-based cooperation because more choices are available. In particular, two main questions should be answered when designing such protocols: when information is 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, which is often obtained using some sort of *lossy* compression, such as Bloom filters [3]. The available possibility ranges 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].

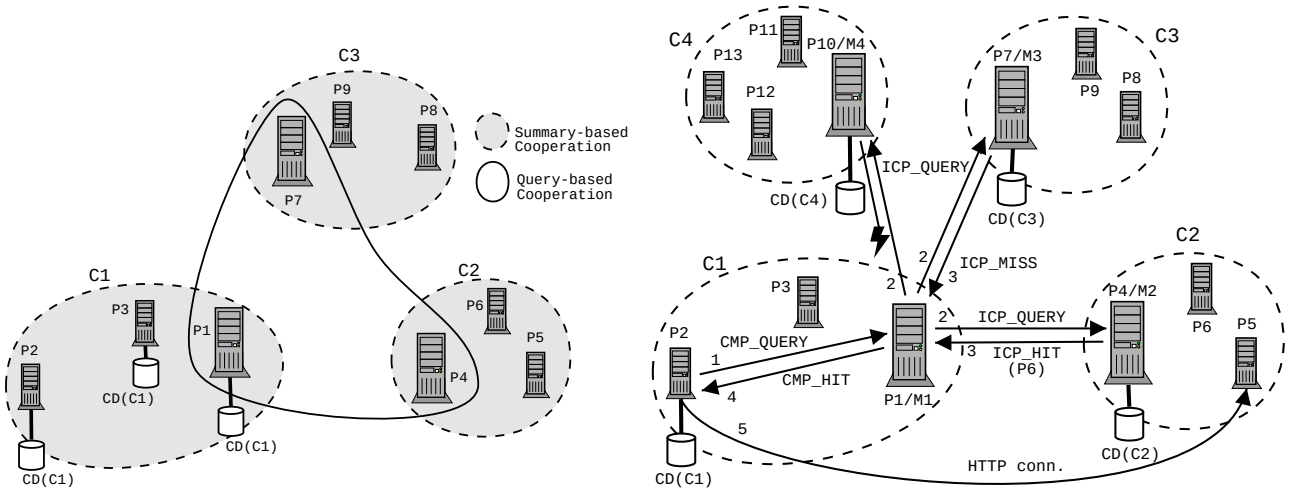
A completely different approach in cooperation is CARP [16] where hashing is used to partition the URL-space between the cache servers. However, this solution is not suitable for a geographic environment where network status changes dynamically in a significative way.

3. 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. We have a *global miss* if the resource has to be retrieved from the origin server, as 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 (first-tier) and a query-based protocol (second-tier) as the mechanisms for intra-cluster and inter-cluster cooperation, respectively. There are some differences between the scheme described in [14] and the prototype proposed here. In particular, we use Cache Digests [12] (instead of Summary Cache as suggested in [14]) as the informed-based protocol and ICP [22] as the query-based protocol. In the Summary-Query scheme, for each cluster we select one server, called *master*, that typically is



(a) Structure

(b) Operational example

Figure 1. Summary-Query architecture

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 mechanism. To show all possible scenarios, this figure includes a new cluster (C_4) which was not present 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.

The peer selection process (used to select a cache server that may hold the requested resource) can use different cooperation protocols, such as ICP and Cache-to-Master Protocol (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. Unlike ICP, CMP is able to report a hit in a third cache server not involved in the actual 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 described cooperation scheme was implemented as a prototype based on Squid 2.4.

4. Novel schemes for hybrid cooperation

4.1. Motivation

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 5. Network researchers using simulation as the performance 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 [17] not including realistic server delays may lead to similar mistakes.

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 two subsections.

4.2. 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 the clients. It is interesting to note that this scheme can be considered as a representative of informed-based architectures that use multiple directories hosted on and 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 an adaptive scheme, where some cache masters are dedicated, while others can accept client requests, depending on the expected level of congestion.

4.3. Query-Summary cooperation

Query-Summary is a quite different alternative to avoid poor performance of the Summary-Query protocol. 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. 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, in Summary-Query the nework resource usage can be unfair. Indeed 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.

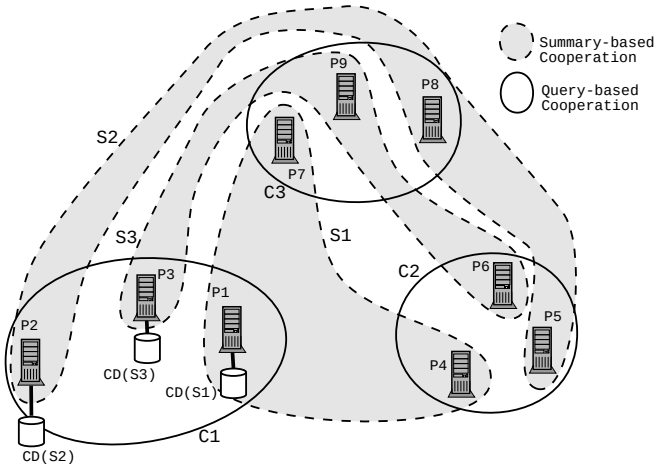


Figure 2. Query-Summary architecture

Figure 2 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 2. 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 [20]. 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.

5. Experimental Results

All proposed 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 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.

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 a synthetic workload generated by Web-Polygraph version 2.5 [13], that intends to capture a "realistic" Internet scenario based on the second cache-off IRCACHE model [8].

The workload 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.

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 typically uses a much higher value: the default is 60 minutes).

5.1. User response time

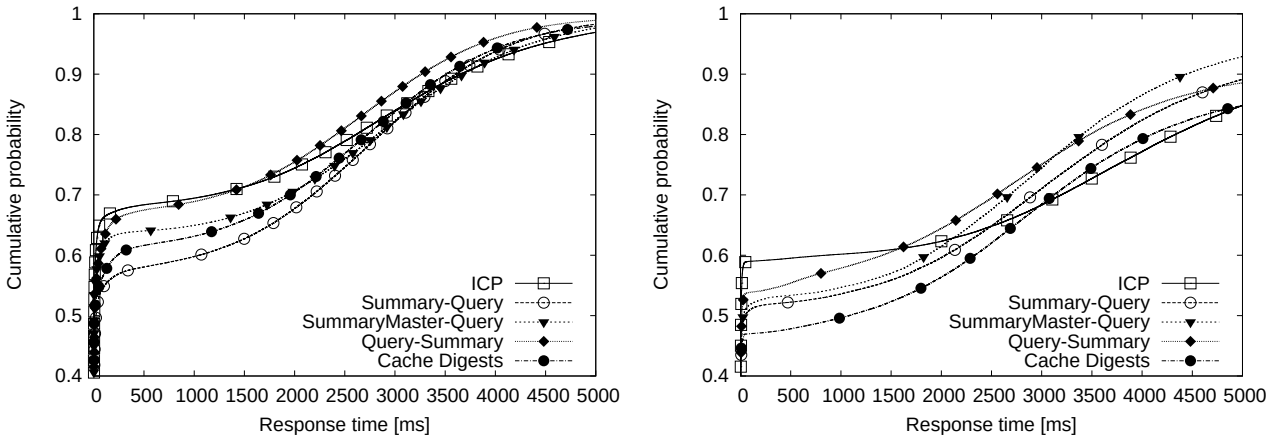
The first set of experiments refers to a geographic testbed consisting of two clusters of five nodes each, connected through some links and a geographical backbone. Ten network hops existed between the clusters. We also installed one Web server in each location. For these experiments, we used the response 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 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.

Figure 3 shows the cumulative probability of the response times of all requests. In particular, Figure 3(a) and 3(b) refer to the light and heavy network traffic, respectively.

It is important to note that 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), as the Bernoullian shape of the response time curves suggest. Higher hit rate means that higher percentages of requests can be served in a short time. Indeed, Figure 3(a) the ICP curve is the highest at the beginning, but it is surpassed by that of Query-Summary (around 1.5 seconds), by SummaryMaster-Query (after 2 seconds) and becomes the worst (after 4 seconds). These results mean that ICP has the lowest 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. These effects become even more evident in other not reported experiments obtained for a workload model characterized by scarce spatial locality and consequent lower hit rates.

From Figure 3(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 the centralized component of any distributed system. 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 the next section.

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



(a) Light network traffi c

(b) Heavy network traffi c

Figure 3. System Response Time

seconds (the highest value), while for Query-Summary it is 3.2 seconds and for SummaryMaster-Query it is 3.6 seconds. Cache Digests, because of its low hit rate tends to be slower in the first part of the curve, but the 90-percentile of its response time (3.6 seconds) is similar to that of the other cooperation protocols:.

As expected, in the case of heavier traffic (Figure 3(b)), the overall performance is reduced because congestion can occur in many places of the distributed system. The percentage of client requests serviced within 100 ms (hit documents) is reduced from 66%-56% to 59%-47%, depending on the cooperation model used. The cache hit rates are affected by similar reductions.

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 policy that is 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 cache hit rate augments the necessity of using crowded links to contact the Web servers.

Looking at the 90-percentile of the response time, we can see that Query-Summary performance is worse than that of SummaryMaster-Query (5.6 vs. 4.4 seconds), which is the best performing protocol with this network condition. This can be explained by considering that in Query-Summary even the 1-st tier lookup process tends to find hits outside the physical clusters. Hence, the retrieval process is potentially more expensive, especially when the network is overloaded.

As a concluding remark, it is interesting to observe that in both cases of light and heavy network load, hybrid cooperation protocols tends to perform better than traditional mechanisms based on pure cooperation, because they can combine the best characteristics of both query- and summary-based protocols.

5.2. Cache hit rates and overheads

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

Table 1 reports the results of cache hit rate and cooperation overhead: the first three columns show the cache hit rates (local, first-tier and global); the last four columns report the traffic overhead due to cooperation, that is indicated as additional number of bytes that must be exchanged for each client request.

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, in our experiments these percentages decrease from 7.5% to 2% of the working set when the cooperating nodes pass from 8 to 30.

It is interesting to note that as the number of peers increases the local cache hit rate is reduced, as the No Cooperation case shows. This effect is not much evident because of the high spatial locality of our workload. A different set of experiments performed with a different workload shows a significative drop in local hit rate (from 40% to 15%) as the number of peer increases. Another noteworthy effect is

Table 1. Cache hit rates (HR) and overheads

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 [$\frac{\text{bytes}}{\text{req.}}$]
ICP							
8	57.22%		69.46%			532.72	66.59
15	52.16%	n/a	68.24%	n/a	n/a	1238.13	82.54
30	45.12%		67.62%			2977.00	99.23
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
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
Cache Digests							
8	31.47%		40.25%			3.67	0.46
15	30.52%	n/a	37.82%	n/a	n/a	5.11	0.34
30	29.14%		37.80%			5.48	0.18
No Cooperation							
8	57.30%		57.30%				
15	52.28%	n/a	52.28%	n/a	n/a	n/a	n/a
30	45.45%		45.45%				

the reduced local hit rate of cooperative protocols with respect to the non cooperative approach. This is caused by the interference of remote hits that reduces the access locality. This effect is particularly evident when the percentage of "private" client requests is high, as in our workload model.

ICP can compensate the reduction of the local hit rate by means of the cooperation mechanism, while Cache Digests is not so effective in dealing with the reduced hit rate: capacity and consistency misses reduces the accuracy of metadata exchanged, thus leading to a high amount of false hits and misses. 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). On the other hand, Query-Summary seems to offer a greater hit rate, especially for higher numbers of nodes achieving an hit rate that is only 5% less than ICP, the best performing protocol. It is useful to remark that the chosen working set tends to overestimate the performance of query-based protocols.

The most interesting differences between the various cooperation approaches can be found in their capacity to distribute overheads due to cooperation. It is evident that the high hit rate of ICP is very expensive. The overhead approaches 3 Kbytes/request with our workload and reaches 4.5 Kbytes/request in other non reported experiments with a different workload. Those results are particularly impressive since the mean resource size is equal to nearly 10 Kbytes. On the other hand, Cache Digests is surprisingly effective in reducing cooperation overhead because of its small amount of exchaged data. Both two-tier 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 shows). However, Query-Summary overhead is almost the double of the SummaryMaster-Query overhead, as shown in column 7 of Table 1. 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.

5.3. Summary of experiments

From the above experimental results we can observe many interesting characteristics about cooperative Web caching.

- ICP is very effective in locating hits, but the cooperation overhead is prohibitively high when many peers are involved.
- Cache Digests is less effective in locating hits, but it is the most efficient cooperation scheme: its overhead is two orders of magnitude lower than that of ICP.

- ICP is very fast in servicing document hits, but it is the slowest scheme in the case of miss documents, hence its performance results are highly sensitive to cache hit rates.
- Two-tier protocols achieve intermediate hit rate and cooperation overhead with respect to ICP and Cache Digests, but the most stable results that seem less dependent on various system characteristics. For example, two-tier protocols guarantee the lowest user response time in both cases of heavy and light network traffic.
- In Summary-Query cooperation, the cluster masters tend to become the system bottleneck much more often than expected and demonstrated by simulations in [15]. In this paper, we avoid the risk of congestion by the so called SummaryMaster-Query cooperation scheme. However, this result leads to two conclusions that can be applied to a wider context than that considered in this paper.

  An under evaluation of CPU costs at the servers for network operations is a big risk for many performance study carried out through (too much) network-oriented simulations.

  This is also a clear message against schemes using centralized directories unless they are hosted on very powerful machines.
- Query-Summary is more effective than SummaryMaster-Query in using fairly the network resources: Query-Summary uses Cache digests on inter-cluster links thus reducing the load on the most critical network resources, while SummaryMaster-Query, on the same links, uses a more expensive ICP-based lookup scheme.

6. Conclusions

This paper offers multiple contributions. We investigate by means of a prototype the benefits in cache hit rate, cooperation overhead 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 [15].

We also propose two novel two-tier cooperation mechanisms that are implemented in Squid-based prototypes. 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 also be appreciated in a geographic environment with heterogeneous links. Moreover, for those protocols, the 90-percentile of their response time is significatively lower than that of pure query- and summary-based mechanisms, especially in the case of heavy network traffic.

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] 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*, Cambridge, MA, Feb. 2002.

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

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

[18] P. Vixie and D. Wessels. Hyper Text Caching Protocol (HTCP/0.0), Jan. 2000. RFC 2756.

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

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

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

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

[23] 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.