SCALABILITY OF COOPERATIVE ALGORITHMS FOR DISTRIBUTED ARCHITECTURES OF PROXY SERVERS

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

Francesca Mazzoni
Dip. Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Modena - Italy
mazzoni.francesca@unimo.it

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

ABSTRACT

Systems consisting of multiple proxy servers are a popular solution to deal with performance and network resource utilization problems related to the growth of the Web numbers. After a first period of prevalent enthusiasm towards cooperating proxy servers, the research community is exploring in a more systematic way the real benefits and limitations of cooperative caching. Hierarchical cooperation has clearly shown its limits. We study the scalability of traditional protocols (e.g., directory-based, query-based) in flat architectures through different performance metrics and experiments using both synthetic workloads and traces. The synthetic workload is also used for sensitivity analysis with respect to various parameters while traces are used for validating our observations in a more realistic scenario.
We show that ICP has a better hit rate than that of Cache Digests, but the latter has a much smaller overhead, thus making the choice between the two protocols a challenge depending on the providers' interest: if the hit rate is the most important parameter, you should certainly choose ICP, while if you are mostly concerned with keeping the overhead low, then your choice should go to Cache Digests. In any case, both protocols show scalability problems when applied to a large number of cooperating cache servers.

KEYWORDS

Web Caching Protocols, Scalability, Performance Evaluation, Distributed Systems.

1 INTRODUCTION

Web caching has evolved as the first way to address Web server and network resource utilization issues related to the growth of HTTP requests. The initial idea of using one proxy server shows very low cache hit rates, even in a context of homogeneous users. *Global caching* or *cooperative caching* architectures are used by public organizations (e.g., IRCache [IRC03], NLANR [NLA03]), Internet Service Providers (ISPs, e.g., AT&T) and third party companies, such as Content Delivery Networks.

After a first period of prevalent enthusiasm towards cooperating proxy servers, the research community is exploring in a more systematic way the real benefits and limitations of cooperation. While there is no doubt that cooperative caching improves performance [KRI01] when applied to a limited working set (this is the motivation of the success of many CDN companies that apply pro-active and cooperative caching only to the content of their customer Web sites), the debate is open over the benefits of cooperation when applied in a transparent way to the entire Web content.

The most important reasons for cooperation among proxy servers are content lookup (*cooperative discovery*), delivery, placement (*cooperative pre-fetching*) and removal (*cooperative replacement*). In this paper, we focus on cooperative discovery schemes, that are traditionally based on two protocol families coming from the theory on distributed systems: *query-based* and *directory-based* protocols. We also consider some delivery issues but do not address cooperative pre-fetching and replacement [KD99, CC02, RL02], that could further improve performance of cooperation.

It is interesting to observe that the debate over the best metric, as well as on the best methodology for evaluating the benefits of cooperative Web caching architectures is still open. Initially, the *cache hit rate* was the main measure used by the studies on cooperative architectures. More recent papers adopt the *user response time* that, by definition, is the measure of interest for the end-user. On the other hand, ISPs and

corporate intranets deploying proxy servers in their networks are interested in the amount of *network traffic reduction*, because it translates to a direct cost reduction. Two main approaches are used depending on the workload used in experiment: many authors use real proxy traces to simulate a more realistic scenario, while other prefer a synthetic workload where parameters can be easily changed, thus achieving a higher flexibility.

The contribution of this paper is twofold: we use both traces and synthetic workload to evaluate scalability of two well known cooperation protocols (namely ICP and Cache Digests) and we perform a wide sensitivity analysis using multiple performance indexes (such as hit rate, overhead and response time) and relating them to many workload characteristics (e.g., client request frequency) as well as different configuration parameters (e.g., number of nodes and cache size). The use of traces allows the simulation of a real scenario, while the synthetic workload offers the flexibility to perform a more flexible sensitivity analysis. In both cases the use of different performance indexes allows a clearer view of the various scalability issues of the two protocols.

The rest of this paper is organized as following. Section 2 describes the cooperation protocols. In Section Workload MODELS we describe the workload models used in the tests. Section Experimental Results presents the experimental results. Section RELATED work presents the related work. Lastly, Section CONCLUSIONs gives some conclusions.

2 COOPERATION PROTOCOLS

There are two main knobs to be handled in the creation of a cooperation scheme among multiple servers: cooperation topology and cooperation protocol.

2.1 Cooperation topology

Cooperation can be established along a vertical direction, namely hierarchical Web caching descending from the Harvest project [CHA95], or along an horizontal direction, namely flat or distributed Web caching [GAD98]. In hierarchical architectures, a cache miss will result in looking for the resource to an upper level cache [YM98]. In flat architectures, every cache is supposed 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. Rodriguez et al. propose a hierarchical structure with cooperation among the nodes at the same level [ROD99]. Tewari et al. investigate the advantages of cooperation using a hierarchical discovery mechanism based on partial cache index distribution with a flat retrieval scheme, where an HTTP connection to a peer is created only in the case of hit detection [TEW99].

Hierarchical architectures follow the idea of hierarchical Internet organization, with local, regional and international network providers. There are so many results describing the drawbacks of pure hierarchical topologies especially for large number of levels in hierarchy [FAN98, TEW99], that in this paper we found more convenient to focus only on flat architectures. This section is devoted to investigating the main characteristics and potential drawbacks of cooperation protocols for document discovery and delivery in a flat topology.

2.2 Discovery protocols in a flat topology

Any cooperation scheme among a set of distributed servers has to define a protocol for exchanging some local state information. In the case of cooperative discovery, this information basically refers to the cache content, although other data can be useful, such as network and/or server load conditions.

Cooperative document discovery has been studied for a while and many mechanisms have been proposed to address the related issues. 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 only occur in response to an explicit request by a peer, and *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.

2.2.1 Query-based protocols

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 document (some implementations keep records of the peer state, so the query is not sent to overloaded or temporarily off-line cache servers). In the positive case, the recipient cache server replies with a hit message, otherwise it may reply with a miss message or not reply at all. This query-based solution was proposed by the Harvest project [CHA95], and then implemented in NetCache and Squid [SQU03]. Both of them use the Internet Cache Protocol (ICP) as their query mechanism.

The limited scalability of ICP has been documented in [WC97b] by the authors of the protocol, while its high overhead for cooperation was one of the main motivations for the proposal of summary-based schemes, such as Summary Cache [FAN98]. In [RW98] Russkov et al. also note that that ICP is slow in handling miss, this motivating the proposal of Cache Digests, another well-known summary-based protocol.

2.2.2 Directory- and summary-based protocols

Directory-based protocols are conceptually more complex than query-based schemes, especially because they include a large class of alternatives, of which the two most important are: the presence of one centralized directory or multiple directories disseminated over the peers; the frequency for communicating a local change to the directory/ies. It is impossible to discuss here all the alternatives that have been the topics of many studies. We limit our analysis to distributed directory-based schemes because it is common opinion that in a geographically distributed system any centralized solution does not scale, the central directory server may represent a bottleneck and a single point of failure, and it does not avoid the query delays during the lookup process.

In a distributed directory-based scheme, each cache server keeps a directory of which URLs are cached in every other peer, and uses the directory as a filter to reduce the number of queries. Distributing the directory among all the cooperating peers avoids the polling of multiple cache servers during the discovery phase, and in the ideal case it 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 relaxation, such as compressed directories (namely, *summary*) and less frequent information exchanges for saving memory space and network bandwidth, respectively. Examples of compression used to reduce the dimension of the transmitted messages are the Bloom filters, used by Summary Cache [FAN98], and Cache Digests [RW98], that offer a form of lossy compression of the cache indexes in which a certain amount of false hits is allowed. These summary-based schemes weaken the consistency of distributed information. However, as a side result of this paper, we have verified that summaries limit network overheads with surprising efficiency, even in case of frequent exchange. For the experiments, we choose Cache Digests as a representative of the summary-based architectures, because of its popularity and its implementation in the Squid software [SQU03].

3 WORKLOAD MODELS

A complete scalability and performance comparison of query-based and summary-based protocols requires experiments that use traces because of their realism, and synthetic workloads because of their flexibility. We outline the characteristics of the two types of workload models.

3.1 Traces

For the trace-driven experiments we consider the IRCache [IRC03] traces collected during 10 working days of July, August and September 2002.

IRCache is a popular cache hierarchy. Their traces are publicly available and widely used for trace-driven simulations and experiments (e.g., [RW98], [DR01]) to the extent that they are a sort of standard for Web caching analysis.

We use the Proxycizer suite [GAD03] to drive the requests to the Squid running proxies. Each proxy server receives the requests stored in a different trace file.

3.2 Synthetic workloads

For synthetic workload experiments, we use Web Polygraph [WPL03] to generate traffic to the caches. It is worth to observe that even our synthetic workload is based on characteristics derived from the second IRCache Cacheoff.

We consider various workload models that intend to capture two "realistic" Internet scenarios. The basic workload is characterized by a high *recurrence* (that is, the probability that a resource is requested more than once), and small inter-arrival times for client requests. This latter characteristic contributes to degrade summary-based cooperation because of capacity and consistency misses. We can anticipate that the workload chosen for direct comparison tends to penalize the summary-based protocol with respect to query-based cooperation. However, in the sensitivity analysis about Cache Digests we also use a different workload with the purpose of showing which parameters make a workload more "Cache Digests- friendly".

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 augments the global cache capacity, but also the size of the working set, because the number of clients is incremented proportionally.

Let us summarize the main characteristics of the workload models.

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% are taken from a "private" set, that is 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. The workload is characterized by a high recurrence, and small inter-arrival times for client requests.

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 coordination. The three main parameters of the Workload 1 that have been modified to get the Workload 2 are as following. 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), hence the popularity distribution has a heavier tail. The working set size keeps growing through the whole experiment hence there is no temporal locality. The public fraction of requests is 100%, which means that every document in the workload is public, so that each resource can be requested by any client.

4 EXPERIMENTAL RESULTS

4.1 Scalability tests

We compare cache hit rates and cooperation overheads for an architecture with an increasing number of proxy servers.

Table 1 summarizes the results. The first column represents the number of cooperating servers, the successive two columns represent local and global hit rates, and the last two columns report the traffic overhead due to cooperation. This last measure is indicated as additional bytes exchanged for each client request.

We can anticipate that the most interesting results for evaluating the cooperation schemes are obtained for Workload 2. Hence, we derive the main conclusions from these experiments and use the results obtained for the other workload models to confirm the results or evidence some differences. We have already observed that increasing the numbers of proxy 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. But the most interesting feature of Workload 2 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 57% to about 16%. Some differences dependent on the cooperation scheme were expected, because a proxy server belonging to a cooperative system receives requests from its clients and other proxy servers. This alters the access patterns and in practice reduces the document locality. From Table 1 we can observe that Cache Digest is not able to face the reduction of the local cache hit rates, especially for higher numbers of peers. On the other hand, ICP compensates the effects of the local hit rate reduction with a global hit rate that is stable and always over 65%.

The motivation for these results is also related to the workload model that tends 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 size of the working set augments. ICP is not affected by the frequency of cache replacements and this explains its good performance.

Let us now pass to consider the overheads due to cooperation. The last two columns of Table 1 show the cooperation overheads that are measured as the ratio between the bytes exchanged for coordination and the number of client connections (hits and misses) received by the proxy servers, that in the last column are normalized by the number of peers.

Table 1. Cache hit rates and overheads for the scalability tests

Number of nodes	Local Hit Rate	Global Hit Rate	Total overhead per request [Bytes/req]	Total overhead per node [Bytes/node]
Workload 1				
		Cache Digests		
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
Workload 2				
		Cache Digests		
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
Traces				
		Cache Digests		
8	21.94	23.68	25.29	3.16
15	24.93	28.83	22.62	1.58
30	22.56	24.71	26.91	0.90
		ICP		
8	25.19	28.32	998.09	124.76
15	29.25	48.10	1852.83	123.52
30	27.43	38.53	4238.54	141.28

As expected, augmenting the number of proxy 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. Its traffic continues to increase with respect to the number of peers in a much faster way than Cache Digests does. It is interesting to note that with 30 nodes the ICP cooperation overhead (traffic generated only to cooperative lookup purposes) reaches 4.5 KB per requests, with an average document size of about 10 KB. On the other hand, Cache Digests shows the lowest cooperation overhead.

For appreciating the stability of the results of the cooperation schemes, we refer to the last column of Table 1, showing the overhead traffic per request relative to each node. It is important that for larger numbers of peers, the relative overhead of Cache Digests tends to diminish, whereas continuous increments

occur for ICP. By comparing the results obtained by means of synthetic workloads to those obtained through the trace analysis, we can see that the ICP overhead shows a similar trend that is, its overhead significatively increases when the number of peers augments, while Cache Digests does not show significative increments.

This study on the protocol sensitivity towards the number of cooperating proxies is new, to the best of our knowledge. The higher overhead of ICP protocol with respect to summary-based ones was also reported in [FAN98], however their study did not take into consideration the overhead sensibility to the number of cooperating proxies.

For what concerns the hit rates, the results obtained through trace-driven experiments confirm that Cache Digests has an hit rate lower than that of ICP. However, it is difficult to evidence any clear trend of the hit rate with respect to the number of nodes. This is a clear limit of a trace-based analysis. As the number of peers increases, more traces are necessary to build the workload model. We recall that each proxy is subject to a different trace, so adding a node requires the addition of a new trace to the workload. The consequence is that the working set of the three experiments (for 8, 15 and 30 nodes) may show, and actually does, different statistical properties. Many previous results based on traces did not clearly evidence this problem. Hence, a synthetic workload model that preserves its features independently of the number of peers seems a really good integration or alternative to the scalability evaluation of large sets of cooperative nodes.

4.2 Sensitivity analysis

In this section we report the results about the sensitivity of ICP and Cache Digests to many parameters. We tested the protocols stability towards the frequency of client requests, the cache size with respect to the working set; for Cache Digests, we also consider its dependency on the digest rebuild period and the digest exchange period.

All experiments show the great stability of ICP vs. the strong dependency of Cache Digests with respect to many (we can say, too many) traffic and system conditions. ICP is quite stable when considering the request rate. We also investigated the consequences of increasing the cache size: ICP cache hit rate augments but this increment tends to saturate (consider that if we increase the cache size by an order of magnitude, the cache hit rate is nearly doubled). As a direct consequence of the increment of the cache hit rate, ICP cooperation overhead diminishes. The cache hit rate of Cache Digests is heavily dependent on many parameters. In particular, it diminishes as the number of peers increases, and even when the working set augments more than the cache capacities.

For these tests we created a third workload, called Workload 3, which is a modified version of the Workload 2. The main differences are as following: all resources are cacheable, their life cycle excludes the possibility of stale hits (that is, in Workload 3, Web objects never expire). Basic Workload 3 has a frequency of 10 requests per second, even if different frequencies are also used for sensitivity analysis.

Some tests carried out through this workload show that the global hit rate decreases from 50% to 33% when the number of cooperative peers passes from 8 to 30 (see the fourth column in Table 2). This instability is also due to the acceleration of the exchanges of objects between caches when the total working set augments more than the cache capacity.

Table 2: Cache hit rate and average overhead per request of Cache Digest (Workload 3)

Number of nodes	Fraction of cache capacity per node	Local Hit Rate	Global Hit Rate	Overhead [B/req]
8	11%	31.83%	50.23%	3.31
15	5%	20.99%	37.01%	4.34
30	3.5%	18.79%	33.80%	5.41

Moreover, we found a (stronger than expected) correlation between the cache hit rate of Cache Digest and the frequency of client request arrivals. In Table 3 we report the impact of the capacity miss on the cache hit rate when 11% of the content can be held in each of the 8 proxy servers. This table shows that the reduction of the cache hit rate when the frequency of client requests increases is significant: from 76% to 50%, when all other parameters are kept constant. The dependency of the Cache Digests hit rate on the number of cooperating peers and on the frequency of client request arrivals are confirmed by all our experiments carried out through traces and synthetic workloads. Actually, they are two effects of the same problem (*out-of-date summary information*) caused by capacity misses and consistency misses, respectively. Capacity misses occur when a directory references an object that the proxy server has previously discarded

to make space to other objects. Larger numbers of cooperative peers augment the working set because more clients are served. If the capacity of each cache node remains the same, the replacement algorithm tends to be activated more often, with consequent higher numbers of capacity misses. In our experiments, each cache can contain from 11% to 3.5% of the entire working set. As shown in the second column of Table 2, the capacity misses due to replacements reduce the hit rate when the peers augment. Consistency misses are caused by the asynchronous propagation of summary contents that further reduces the accuracy of the directories, especially when the frequency of client request arrivals is much greater than that used for summary refresh.

On the positive side for the summary-based protocols, we have that the overhead for cooperation is really low and almost independent of the number of peers: the traffic generated for coordination is even three orders of magnitude less than that caused by ICP. Moreover, this cooperation overhead is quite stable with respect many parameters: it is mainly a function of the cache capacity, because increasing the capacity augments the size of the digests being exchanged.

Table 3: Global Hit Rate of Cache Digest as a function of the frequency of client requests

Client request frequency [req/s]	Global Hit Rate
10	50%
5	72%
2	74%
1	76%

We also studied the sensitivity of Cache Digests to other parameters, such as the cache rebuild period, the digest exchange period and the cache size. We do not present the detailed results because of space reasons. For example, by using 8 nodes, and three digest rebuild period values (300, 60 and 20 seconds) we have the following results: changing from 300 to 60 seconds brings to a sensible hit rate increase; the next step (from 60 to 20 seconds) brings very small advantages to the cache hit rates, but it tends to augment the overhead of Cache Digests. Hence, we can conclude that, for this workload, an interval of 60 seconds is a good trade-off between hit rate and overhead. We think it is important to notice that Squid allows you to change, without modifications to the source code, only the digest rebuild period, which is related to the digest exchange period, but not in a linear way. For this reason we also modified the Squid source code to verify whether a modification of the digest exchange period could help in having a better hit rate. As expected, diminishing this interval leads to an increment of the hit rate, but not in a significative way. Moreover, its cooperation overhead increases due to the more frequent exchange of the digests.

Lastly, we investigated the consequences of increasing the cache size for Cache Digests: its hit rate augments (increasing the cache size by an order of magnitude, the cache hit rates is nearly doubled), but even in this case, it tends to saturate. Moreover, its overhead increases, due to the bigger sizes of the exchanged cache digests, although in a much lower way than that observed for ICP.

4.3 Response time tests

Geographical tests were made by setting up two clusters of Web proxies in two locations, separated by ten network hops. We made the tests on a busy day and repeated them on Sunday, to check possible changes. There is a clear correlation between the cache hit rate and the user response time: requests are serviced very early or after a large amount of time. 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). ICP shows the best response times for hit documents, but poorest results in the case of global misses because it has to wait for the slowest peer.

The conclusions can be better appreciated by evaluating the 95-percentile of the response time. In the case of light load, the 95-percentile for ICP is 4.5 seconds (the highest value) and for Cache Digests is 4.1 seconds.

When heavy load is placed on the network, congestion can occur in many places. The overall performance of the cooperative system is reduced (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.

Also the 95-percentile of the user response time augments: ICP shows once again the best performance (10.5 seconds). Cache Digests is the worst cooperation approach in terms of 95-percentile (13.2 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

Cooperative Web caching has been studied for a while and many cooperation techniques have been proposed. Usually, each cooperation scheme is validated to demonstrate its benefits. However, most of those performance analysis focus on a single performance metric, without wide sensitivity analysis.

The definition of measures of scalability for Web caching and cooperative Web caching remains an important topic in performance evaluation. Unlike other papers, we consider the main metrics appeared in literature: cache hit rate, response time and overhead for cooperation. In most previous studies, object or byte hit rate is the main metric used to evaluate performance of a caching architecture. More recently, several authors have pointed out the importance of considering other performance metrics. For example Fan et al. [FAN98] introduce network overhead as a measure for system scalability, and the impact of cooperation on central memory and CPU usage. However, present architectures do not seem anymore to represent a bottleneck for proxy servers with the exception of mass storage size for nodes holding large resources such as video data [LEE02]. The user response time is another important measure of cache performance. For example Russkov et al. [RW98] use this parameter to demonstrate that Cache Digests is preferable to ICP. Dikes and Robbins [DR01] study the maximum speedup achievable by cooperative caching and demonstrate that user response time reduction is related to cache hit rate. Moreover, they find that cooperative caching allows a more efficient usage of network resources by avoiding congestion and leading to less variable response time.

ICP is perhaps the most widely studied cooperation protocol. Traces are used to evaluate scalability through overhead and response time respectively in [FAN98] and [RW98].

There are much less studies on directory- and summary-based cooperation. The most significative paper on Cache Digests performance is [RW98]. To the best of our knowledge, no wide sensitivity analysis on different parameters has been done for this protocol.

We consider important to observe that using also a synthetic workload (instead of relying only on trace-driven experiments) allows us the highest flexibility on the choices of the workload characteristics. We carry out important sensitivity analysis that are prevented to the large majority of papers based only on trace-driven tests. Indeed, there are few sensitivity analysis of cooperative caching performance to workload characteristics. Williamson et al. focus on traditional proxy caching [WB01], while some results on cooperative Web caching is in [LEE02] and [DP02].

6 CONCLUSIONS

We evaluate and compare the performance of two widely used cooperation protocol for Web caching: ICP and Cache Digests. For our experiments we use traces and synthetic workloads. Synthetic workloads offer us the flexibility to perform sensitivity analysis to single parameters of the workload, while traces allow a validation of some observations in a realistic scenario and a better comparison with existing literature. We evaluate multiple performance indicators, such as cache hit rate, overhead and response time, relating many of them to workload characteristics (e.g., client request frequency) or configuration parameters (e.g., number of nodes).

All results tend to show that query-based protocols, such as ICP, and a summary-based protocols, such as Cache Digests, have severe scalability problems when applied to a large number of cooperative proxies. In particular, query-based protocols seem very effective in resource discovery, however their high cache hit rates are affected by cooperation overheads that grow more than linearly as the number of peers increases. The tests in a geographic environment have evidenced that the query mechanism has low response time in the case of hits, but it is much slower than Cache Digests in handling misses. We can suggest to use the ICP protocol only when cooperating caches have very high speed network interconnections.

Cache Digests causes a negligible overhead for cooperation, but its cache hit rates are very sensitive to many workload and architecture parameters. In particular, when the rate of client requests is high and/or there is a frequent object replacement, the accuracy of the digests is very low. This leads to poor cache hit rates, much lower than that of ICP. Even in a geographic environment, Cache Digests performance is mainly

affected by its low hit rates. We suggest to have the biggest cache size as you can in order to avoid capacity misses and to have quite a small digest exchange and rebuild period, so to have less consistency misses as possible. Even for Cache Digests, fast network interconnections among the cooperating caches lead to much better performance.

REFERENCES

[CC02] Chang, C. Y. and Chen, M. S., 2002. A new cache replacement algorithm for the integration of web caching and prefetching. *Proceedings of ACM 11th Int. Conf. on Information and Knowledge Management*, McLean, Virginia, USA, pp. 632 - 634

[CHA95] Chankhunthod A. et al., 1995. A hierarchical internet object cache. *Proceedings of Usenix Annual Technical Conference 1995*,. NewOrleans, Louisiana, USA

[DP02] Du, P. and,Subhlok, J., 2002 Evaluation of Performance of Cooperative Web Caching with Web Polygraph, 7th International Workshop on Web Content Caching and Distribution (WCW), Boulder, CO, USA

[DR01] Dykes, S. G. and Robbins K. A., 2001. A viability analysis of cooperative proxy caching. *Proceedings of IEEE Infocom*,. Anchorage, Alaska, USA, pp. 1205-1214

[FAN98] Fan, L. et al., 1998. Summary cache: A scalable wide-area web cache sharing protocol. *Proceedings of ACM SIGCOMM '98*,. Vancouver, B.C., Canada, pp. 254-265

[GAD98] Gadde, S. et al., 1998, A Taste of Crispy Squid. *Proceedings of the Workshop on Internet Server Performance (WISP'98).* Madison, WI, pp. 129-136

[GAD03] Gadde, S. The Proxycizer suite, 2003: http://www.cs.duke.edu/ari/cisi/Proxycizer/

[IRC03] The IRCache Project, 2003 http://www.ircache.net/

[KD99] Koropolu, M. and Dahlin, M., 1999. Coordinated placement and replacement for large-scale distributed caches. *Proceedings of IEEE Workshop on Internet Applications*, San Jose, CA, USA

[KRI01] Krishnamurthy, B. et al., 2001. On the use and performance of content distribution networks. *Proceedings of Internet Measurement* Workshop. ACM SIGCOMM, San Diego, CA, USA .

[LEE02] Lee, K. W. et al., 2002. On the sensitivity of cooperative caching performance to workload and network characteristics. *Proceedings of ACM Sigmetrics '02*, Marina del Rey, California, USA, pp. 268-269

[NLA03] NLANR. National laboratory for applied network research, 2003. http://www.nlanr.net.

[RL02] Ramaswamy, L and Liu, L, 2002. A new document placement scheme for cooperative web caching. *Proceedings of IEEE 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, pp. 95 – 103

[ROD99] Rodriguez, P. et al., 1999. Web caching architectures: hierarchical and distributed caching. *Proceedings of Web Caching Workshop (WCW'99)*, San Diego, California, USA

[RW98] Rousskov, A. and Wessels, D., 1998. Cache digests. *Computer Networks and ISDN Systems*, vol. 30, num 22-23, pp. 2155 - 2168.

[SQU03] The Squid Web Proxy Cache, 2003 http://www.squid-cache.org

[TEW99] Tewari, R. et al., 1999. Beyond hierarchies: design considerations for distributed caching on the internet. *Proceedings of 19th International Conference on Distributed Computing Systems*. IEEE

[WB01] Williamson, C. and Busari, M., 2001. On the sensitivity of web proxy cache performance to workload characteristics. *Proceedings of IEEE Infocom '01*,.Anchorage, Alaska, USA

[WC97a] Wessels, D. and Claffy, K., 1997. RFC 2187: Application of internet cache protocol (icp), version 2

[WC97b] Wessels, D. and Claffy, K., 1997. Icp and the squid web cache, http://www.nlanr.net/~wessels/Papers/icp-squid.ps

[WPL03] Web Polygraph, 2003. http://www.web-polygraph.org/

[YM98] Yu, P. S. and MacNair E. A., 1998. Performance study of a collaborative method for hierarchical caching in proxy servers. *Computer Networks and ISDN Systems*, pp. 215-224