Building a Flexible Web Caching System

Víctor J. Sosa Sosa, Gabriel González S.,
Centro Nacional de Investigación y Desarrollo Tecnológico
Interior Internado Palmira S/N, Cuernavaca, Morelos, México. 62490
{vjsosa, gabriel}@cenidet.edu.mx

Leandro Navarro
Universitat Politècnica de Catalunya (UPC)
Jordi Girona, 1-3, D6-105, Campus Nord .
Barcelona, Spain. E-08034
leandro@ac.upc.es

Abstract

Web caching is a technology that has demonstrated to improve traffic on the Internet. To find out how to implement a Web caching architecture that assures improvements is not an easy task. The problem is more difficult when we are interested in deploying a distributed and cooperative Web caching system. We have found that some cooperative Web caching architectures could be unviable when changes on the network environment appear. This situation suggests that a cooperative Web caching system could get worst access to Web objects. However in this paper we present an architecture that combines the best of several Web caching configurations that we have previously analyzed. Our architecture gives basic ideas for implementing a cooperative Web caching system using groups of HTTP proxy servers which can improve access to remote Web objects regardless of the changes that might occur on the network environment (changes that could produce modifications in Web object validation policies and/or types of caching communication).

1. Introduction

The idea behind Web caching consists in getting Web objects close to clients at a low cost. Cooperating proxy caches are a group of caches that share cached objects and collaborate with each other to do the same work as a single Web cache. The benefits of having a cooperative caching system has been analyzed in [12],[6],[1]. Basically, the construction of a cooperative Web cache system requires the analysis of four major topics. They are briefly described next:

Cooperative caching system organization: How to define a cache topology.

Hierarchy: Caches are located at different network levels. In most cases it is assumed that inferior levels in the hierarchy have better quality of service. They have a parent-son relationship. A son cache is located at inferior levels in the hierarchy, and when a son cache needs a Web object, the son cache asks its parent cache for it. The request goes up in the hierarchy until finding the Web object needed in a parent cache or in the original Web server.

Mesh (distributed): There are no intermediate caches defined by levels, rather there is a single level of caches where they can cooperate to serve the requests generated by clients.

Hybrid (mesh/hierarchy): A combination of hierarchy and mesh.

Web caching communication: How caches are going to communicate each other. We consider three processes that are involved in Web caching communication: discovery, delivery, and dissemination.

Discovery. How do caches find the Web objects? There are three major approaches: Exhausted query: asking for a requested object to all sibling caches using a protocol like ICP (Intercache Communication Protocol). Using digest: Digest can be interchanged using methodologies such as: Peer to peer or by a hierarchy. Using hashing: cache objects can be located in proxy caches defined by a hash function.

Delivery. How do caches deliver pages to clients? It could be using direct connection between the cache containing the page and the client, delivering copies using a cache hierarchy, or delivering copies using a cache mesh.

Dissemination. Delivery of Web objects initiated by original servers.

Consistency strategies: How caches keep "fresh" cached objects.

Expire. Using predefined expiration dates on Web pages.

TTL0. Verifying consistency every time a hit occurs (time to live is zero, ttl=0).

TTLA. Time to live is assigned based on the elapsed time since the last request.

Invalidation. Original server sends an invalidation message to caches when a Web object update occurs. Usually this messages is sent via multicast (invalidation multicast). Some variants of multicast invalidation or a mix of all previous strategies.

Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)
0-7695-1915-6/03 $17.00 © 2003 IEEE

Workload behavior: Behavior of workload can be treated depending on the participating element: Clients, Proxy-Caches and Servers. In this case we focus our attention on proxy-caches workload.
The goal of this work is to show many options for building a viable Web caching system based on many analyses of different Web caching configurations. We also propose a Web caching architecture that takes the best of several Web caching configuration. The principal goal of this Web caching architecture is to avoid problems that could make it unviable when the network behavior changes.

2. Architecture description

This architecture is based on our previous research [8],[9],[10], which encompasses several studies of Web cache systems (workloads, consistency strategies, Web caching communication). The main characteristics of our architecture are:
1) Web caching organization. Considering the obtained results in [8],[9],[10], we propose a caching organization of hybrid type (three level hierarchy in long latencies, where each level has a mesh, Fig. 1). It has been demonstrated in [8],[6],[12] that a hybrid organization offers better results as for access time, bandwidth consumption in a wide area network (WAN), and scaling support. Likewise, a three level hierarchy has been widely accepted [4],[3],[7],[5] because it does not emphasize the cost of store-and-forward process.
2) Caching communication. As we have mentioned, the communication in a cooperative caching system takes places in several processes (discovery, delivery, and dissemination).

Discovery process: This process allows for the locating of requested Web objects within the cooperative caching system. A very typical mechanism is the exhaustive query by using ICP messages. However, exhaustive query will not be used in our architecture, due to the high bandwidth consumption that it requires [2]. A mechanism for the construction of directories (metadata) will be used instead. The construction of such directories will be done dynamically. When a parent cache receives a request for a Web object, immediately after the requested Web object has been sent to its requester cache, a multicast message is sent to all of the parent cache descendents to notify them the Web objects that have been requested by the requester cache. This lets the lower levels of the hierarchy know the location of these Web objects for further requests. The caches that will receive such notification are only those caches that belong to that particular branch in the lower levels of the hierarchy.

Delivery process: Clients obtain a copy of the requested Web object through the mesh (caches of the same hierarchy level –branch-). Those copies have arrived to the mesh following a route from the original server to the cache through the hierarchy levels.

Dissemination process: This process takes advantage of the multicast invalidation mechanism to push updated Web objects that have been previously requested by the lower level caches in the hierarchy. This process will be fully explained next.

3) Consistency mechanism. The consistency mechanism to be used in this architecture is *multicast invalidation,* and it uses what we call *life signals*. The parent cache sends a message (multicast message -life signal-) every minute to the lower level caches to indicate that it is alive (on line). If a Web object is modified in the original server (one of those Web objects previously requested), a message is sent to the lower level caches within the same mesh, to let them know that such Web object is no longer valid, and that an updated copy of the Web object should be obtained the next time the Web object is requested. The same *life signal* is used to inform and prevent those caches that did not contain the Web object from trying to access such a Web object from an invalid copy.

Dissemination strategy (pushing) through the consistency mechanism. We have a dissemination strategy (pushing) where the servers (or parent caches in a hierarchy) multicast the most popular Web objects (that have been updated by the original server) to the caches, before being requested once again. In this architecture, no global decision is taken concerning the multicasting of a Web object through the system. Conversely, each cache and the original server make their own decision on whether the dissemination should be made or not. To do this, each cache and the original server keep an access counter (Xp) for each Web object which in the beginning is set to 0, along with a dissemination bit that will indicate whether the Web object must be disseminated or not. If the dissemination bit is set to 1, that will indicate that the cache has to disseminate the updated Web object when the cache issues a life signal to its lower level caches. The heuristics of this strategy use three positive variables: υ,β,δ. If a cache receives an invalidation message for page P, then the next calculation takes place: $X_P = X_P - \beta$. if the cache receives a request for page P, then we calculate: $X_P = X_P + \delta$. If X_P exceeds a threshold υ ($X_P > \upsilon$), then the dissemination bit is set to 1, otherwise a zero is assigned. Additionally, the lower level caches send a message to their parent caches each time the page is read. This is done to keep a knowledge about all read pages within that branch of hierarchy. The fixed values used within our architecture are $\upsilon=8$, $\beta=1$, y $\delta=2$. The approach that we followed of taking a 1 for each invalidation and of adding a 2 for each request, is due to the popularity studies made with some proxi-caches logs that are strongly close to the Zipf distribution. The Zipf distribution says that the number of requests for a page (R) is related to its popularity in this way:

$$R(i) = \frac{\Omega}{i^{\alpha}}$$

where the exponent α reflects the popularity skew between one Web object and another, and the constant Ω defines an approximation to the number of requests for the most popular Web object represented by $i = 1$. For a better understanding, lets assume that $\alpha = 1$ and that $\Omega = 100$, then the most popular Web object (in other words, the one with the position $i = 1$) will have 100 requests, the Web object next to it in popularity will have 50 ($i = 2$), and the next ($i = 3$) will be close to 30 and so forth. All this gives us an idea that the popularity of Web objects follows a sequence of 2 to 1 for each position in the popularity index. This helps us to define constants β and δ. The υ value comes from studies made in this work, along with heuristic analysis made in [13].

3. How this web caching architecture was defined

This Web caching architecture was substantially obtained using a simulator developed during the time of this work [10]. This simulator is mainly an extension of the Network Simulator (NS) [11]. The fundamental research that helps us to define this architecture could be found in our work [8], which basically consisted in analyzing the most studied ways to organize cooperative Web caches like: hierarchies, mesh and hybrids, combined with the most studied Web caching consistency strategies. In this work we tried the following consistency mechanisms: TTL0, TTLA, M, and some variants of multicast invalidation mechanism: PSM and APM. PSM (pushing selective multicast) only pushes the updated Web object when the number of requests exceeds a threshold defined by υ (as mentioned in the previous section). APM (always pushing multicast) pushes a Web object whenever it is updated. The configurations mentioned above were implemented in our simulator. We use workloads that were obtained from a cache located at the Spanish academic network backbone managed by the Center of Super Computing of Catalonia (CeSCa). The simulation stops when it finishes replaying all the workload. Table 1 describes the workload. The Web caching architectures that give us the best results in [8] were the base for our new Web caching architecture discussed here .

Table 1. Workload characteristics.

Number of requests	3,089,592
Number of Web objects	212,352
Number of clients (approx.)	11,765
Average requests per second	21
Transferred bytes	30GB
Duration	2 days

4. How this architecture works

This section describes how this architecture works. The Web caching system starts when a client sends a request to its Web proxy cache (the first cache that is contacted by a client is called client-cache). The client-cache verifies if it can respond to the request. If so, the client-cache sends the requested Web object to the client and the operation is finished. Otherwise, the client-cache checks if the requested object is in a sibling cache. A sibling cache is a cache connected by a mesh at the same branch in a cache hierarchy (see Fig. 1). Every cache sibling has a digest where the cached objects in a hierarchy branch are registered. If the requested Web object is in a sibling cache (a cache knows that by looking at its digest), the cache forwards the request to the cache sibling. A copy of the requested Web object travels to the end client through the client-cache which keeps a copy of the Web object. If the requested Web object is not in any cache sibling then the request is forwarded to the parent cache. The same process is repeated at each hierarchy level until the requested Web object is found in a parent cache or in the original server (if the requested Web object is not found in the Web caching infrastructure). Every parent cache keeps a list of pairs {requester_son_cache, requested_Web_object} which is sent to all its son caches piggybacking the life signal used in the consistency mechanism explained before. Every cache receiving that list includes it in its digest. In that way every son cache is building its own digest. The list is removed from the parent cache right after the list has been sent. This reduces control overhead in parent caches. As we have said before, every cache digest only has references to Web object copies stored at the same branch (mesh which the cache belongs to) in the hierarchy. Likewise, when a Web object is modified in the original server, the server keeps a list of modified_Web_objects to be sent piggybacking the life signal used in the consistency mechanism. Figure 1 shows an example of our architecture. We can see in Figure 1 with a double line the path that the life signal will take.

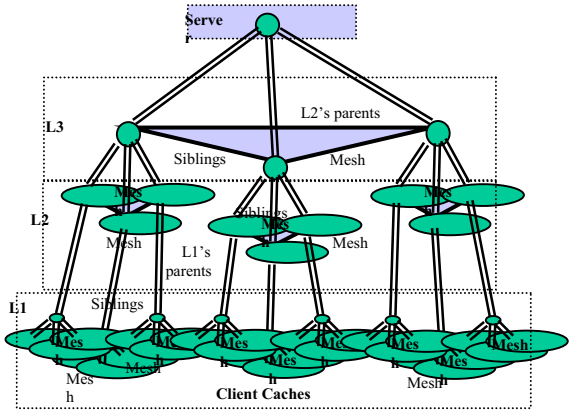


Figure 1. A possible scenario using the Web caching architecture

The life signal is an indicator for son caches that the parent cache is on line (alive). If a son cache does not receive five consecutive life signals, then the son cache presumes that

its parent cache is dead. After that, the son cache invalidates all the Web objects received by its parent cache. We allow five missed messages because we wanted to consider some high network congestion. Every parent cache (at every hierarchy level) will forward (multicast) the invalidation message with the life signal if one of its son caches has the invalidated Web object, otherwise the parent cache will forward only the life signal. If a Web object is modified in the original server and the push mechanism is activated (pushing bit = 1), then the new Web object will be piggybacked on the life signal. If a parent cache has not been requested with that Web object(s), the parent cache will not forward the Web object(s) but only the life signal.

5. Analysis of this architecture

This section shows some comparative analyses of the Web caching architecture suggested in this paper. Table 2 summarizes the simulation scenarios. These simulation scenarios use the workload described in table 1.

Table 2. Scenarios for comparing Web caching architectures

Identifiers	Description
JTTLA0, JTTLA, JM, JPSM, JAPM	Hierarchical cooperation architecture (J) using the following consistency mechanism: TTLA0, TTLA, M, PSM, APM
DTTLA0, DTTLA, DM, DPSM, DAPM	Distributed cooperation architecture (D) using the following consistency mechanisms: TTLA0, TTLA, M, PSM, APM
H1TTLA0, H1TTLA, H1M, H1PSM, H1APM	Hybrid cooperation architecture 1 (H1) using the following consistency mechanism: TTLA0, TTLA, M, PSM, APM
H2TTLA0, H2TTLA, H2M, H2PSM, H2APM	Hybrid cooperation architecture 2 (H2) using the following consistency mechanism: TTLA0, TTLA, M, PSM, APM
ADDTTA0, ADDTTLA, ADDM,ADDPSM, ADDPM	Web Objects Distribution architecture (ADD), suggested in this paper, using the following consistency mechanisms: TTLA0, TTLA, M, PSM, APM

The first that we can see in this analysis is the perceived response time by clients. Figure 2 shows how the consistency mechanisms affect every cooperative caching organization in several ways along the workload on this simulation scenario (CESCA's cooperative Web caching system). We can see that distributed caching cooperation architecture in this context presents the best results. It is important to notice that slightly lower in this architecture we can see the ADD architecture which shows better results than the others. It was not relevant which consistency mechanism was used. It is interesting to notice that when we have TTLA as a consistency mechanism implemented on a hybrid cooperative Web caching architecture (H2), response times are good. However, if the consistency mechanism is changed in this architecture, the response time increases considerably..

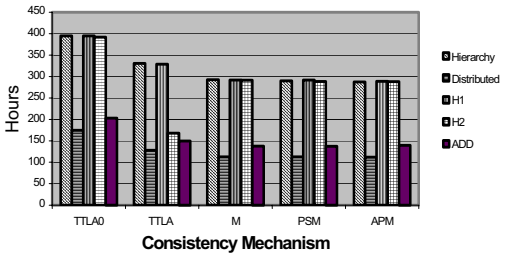
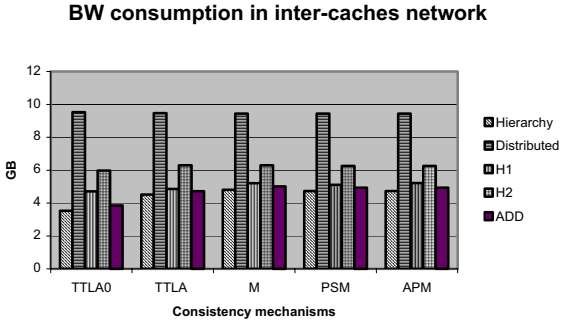
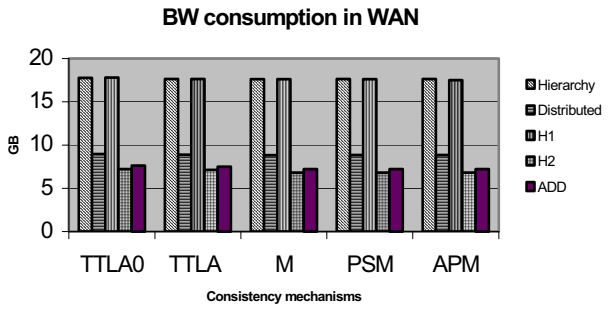


Figure 2. Total response time perceived by clients

Figure 3 shows the bandwidth (BW) consumption of the inter-cache network and traffic generated to the wide area network (WAN). It is important to see both traffics in a separate way because we can see where the bottlenecks are (if any), and to be aware if our HTTP traffic is yielding some problems to another type of traffic inside and outside the inter-cache network.





and outside of the inter-cache network

Distributed architectures have the highest inter-cache bandwidth consumption. Conversely, hierarchies have the lowest inter-cache bandwidth consumption independently of which consistency mechanism is being used. We can see in Figure 3 that TTLA0 is the consistency mechanism which consumes less inter-cache traffic. TTLA0 has a strong dependence on Web servers outside inter-cache network because it validates every request that caches receive for a stored Web object. TTLA0 produces high WAN traffic. WAN traffic is important because it could be the reason for variability in client-perceived response time. If we generate less impact in WAN traffic then it is

possible to prevent response time peaks, preventing clients from perceiving pathological response time. Figure 3 shows ADD architecture with competitive bandwidth consumption for both inter-cache links and WAN links. Every consistency mechanism implemented in every cooperative Web caching architecture was configured according to mentioned parameters in every original cooperative Web caching architecture proposal. In Figure 4 the number of received "stale" Web objects by clients is compared. As we can see, if our interest is getting strong consistency, the consistency mechanism to be chosen is TTLA0 (it is not important which type of cache organization is implemented). PSM will be the next option.

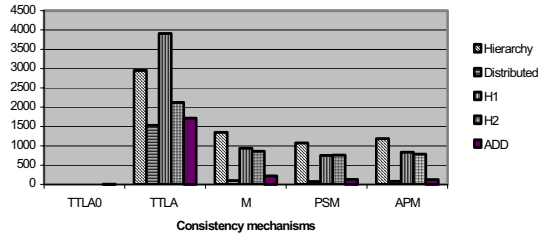


Figure 4. Number of "stale" Web objects received by clients

We measure the benefit degree (if any) that is produced by the cooperative caching architecture and its consistency mechanism compared with a system that does not use caches. It was taken the sum of the response times obtained reproducing the trace file in each one of the systems as a measure of comparison.

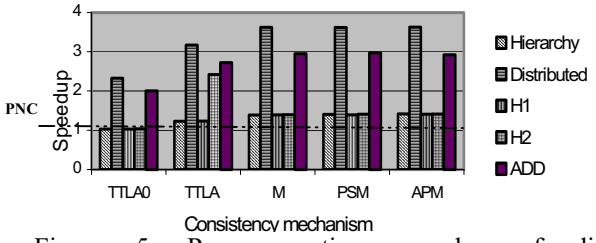


Figure 5. Response time speedup of different configurations (PNC: Proxy No Cache

Figure 5 shows the degree of gain (speedup) obtained by each cooperative architecture and its respective consistency mechanism. We have called the architecture that does not use caches "Proxies No Caches" (PNC). A speedup equal to 1 indicates that the cooperative caching architecture and the validation mechanism that it uses is not generating gains. Speedups greater than 1 mean the cooperative caching architecture obtains some gains. As we can see in Figure 5, the best results are in distributed and ADD architectures, no matter which consistency mechanism is implemented. They have speedups higher than 2.5 and 3.5. In Figure 6 all the architectures reviewed in this work appear contrasting speedups with the bandwidth consumption that they generate. The proximity to the ideal mark represents greater benefit with smaller cost. ADDPSM, ADDAPM, DAPM, DTTLA y H2M configurations are the best alternatives when the interest is focused on the network infrastructure. We have done more experiments scaling all caching architectures to more caches (for more than 20 caches, these experiments are not included in this paper). If we scale (including more caches) the cooperative caching architecture, we can see that distributed caching architecture was the most affected in terms of high response time and decreasing speedup. When a distributed caching architecture grows, the number of inter-cache messages grows linearly. That means that the bandwidth consumption grows as well, and the inter-cache performance goes down.

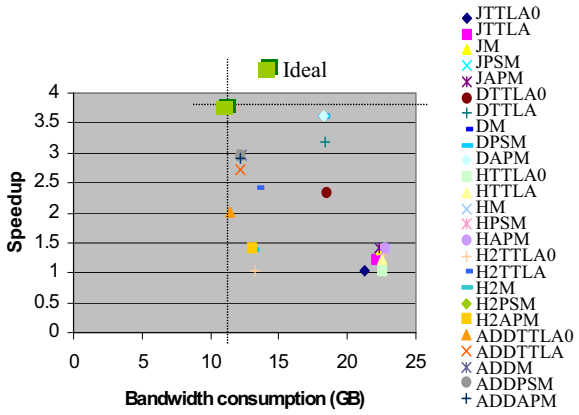


Figure 6. Speedup vs. BW consumption

6. Conclusions

This paper presents a Web object distribution architecture which is resilient. That means, if the network behavior changes, and there is a need to make some changes to the Web caching architecture to make up the situation, the changes could be done and Web caching architecture still offers benefits without considerably affecting the network resources. The approach that we have implemented was to evaluate the most popular cooperative cache architectures combined with the most used consistency mechanisms. After analyzing the results of these evaluations, we have built a Web caching architecture that we called ADD. This caching architecture takes advantage of the best features detected in other caching architectures. That is why this caching architecture obtains better performances even if some changes in configuration have to be done. ADD architecture is based on a cooperative cache hybrid organization including an inter-cache Web object discovery mechanism based on directories (digest). The ideal consistency mechanism for this architecture is invalidation multicast with life signaling to keep the state of parent-son cache links. If some changes in network behavior occur, it could be necessary to change the

consistency mechanism. That situation is not going to be all that important because we trust the cache system will still offer some benefits. This is a very important advantage of this architecture. During this work, we have developed a reliable simulator which is a useful and dependable tool for an a priori evaluation of Web caching architectures and is a good contribution of this paper.

References

[1]S.G. Dykes, C.L., K.A. Robbins, and C.L. Jeffery, "A Viability Analysis of Cooperative Proxy Caching", in *Proc. of the IEEE Infocom 2001*

[2]"Internet Cache Protocol (ICP) version2" available at: ftp://ftp.rediris.es/docs/rfc/21xx/2186

[3]Korea National Cache, available at: http://cache.kaist.ac.kr

[4]National Laboratory for Applied Network Research (NLANR), "Ircache project", available on line at http://ircache.nlanr.net/

[5]Spanish academic network (Red Iris) available at: http://www.rediris.es/

[6]P. Rodriguez, C. Spanner, and E. W. Biersack, "Web Caching Architectures: Hierarchical and Distributed Caching". *4th International Web Caching Workshop*, San Diego, USA. 31st March –2nd April, 1999

[7]N. G. Smith, "The UK national Web Cache – The state of the art", *Computer Networks and ISDN System*, 28:1407-1414, 1996

[8]V. J. Sosa, L. Navarro, "Influence of the Document Replication/Validation Methods on Cooperative Web Proxy Caching Architectures". Communication Network and Distributed Systems Modeling and Simulation Conference CNDS'02 in WMC'02. Pags. 238-245. ISBN: 1-56555-244-X. San Antonio, Tx. USA

[9]V.J.Sosa, L. Navarro, "A New Environment for Web Caching and Replication Study", Workshop of distributed and parallel systems (WSDP'00). University of Santiago. Nov. 13-18 2000. Chile. ISBN 956-7069-53-0. CD-ROM

[10]R. Tewari, M. Dahlin, H. M. Yin, and J. S. Kay, "Design considerations for distributed caching on the internet", in Proc. of the Int'l. Conf. On Distributed Computing Systems (ICDS'99)

[11]Virtual InterNetwork Testbed Project. Available at: http://netWeb.usc.edu/vint/

[12]A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative Web Proxy caching", in Proc. of the 17th ACM Symp. On Operating Systems Principles, Dec. 1999

[13]H. Yu, L. Breslau, and S. Shenker, "A Scalable Web Cache Consistency Architecture". SIGCOMM99. volume 29, number 4, October 1999. http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session5-1.html