Some Recommendations on Building Proxy Caching Service

Andrey Naumenko
ICA-EPFL, Swiss Federal Institute of Technology, Lausanne CH-1015.
E-mail: andrey.naumenko@epfl.ch

Abstract: In this paper, we present our research on WWW caching proxies. We drew some key conclusions on the service properties and caching policies, according to the results of our analysis of caching proxies. These conclusions include our recommendations on cache configuration and illustrate how choosing cache scenarios depends on the different requirements requested by the cache manager and different network conditions.

1. Introduction

In response to the growing demand for web caching services, much research has been made on WWW caching and replications. Today, there are several technological possibilities available to support caching service. However, as research continues to improve these technologies, there has not been enough analysis to provide recommendations for making a concrete choice that depends on the particular interest and conditions of cache exploitation.

In this paper we will look at some of the possible variants for building a cache system, compare their performances under different conditions, and recommend the best options for several typical cases of cache usage requirements.

We will review the main functionalities required from caching proxies and concentrate on the protocols for their support in the Section 2. Section 3 is dedicated to the configuration policy for caching proxies; we will analyze criteria and make recommendations on it. In section 4 we discuss the ideas on the policy of caching. Possible areas for future research are highlighted in the Section 5, and our conclusions are presented in the Section 6.

2. Web Caching Review

2.1. Proxy Caching

The advantages of proxy caching are well known; the most valuable are the reduction of Internet traffic and the decrease of response time for WWW users. The most important functions that are required from modern caching proxies involve support for hierarchical caches, distributed caching, and the auto pre-fetching of the objects, which have expired in the cache or active caching. Hierarchical caches make possible for a downstream proxy to forward requests to upstream proxies rather than to the Internet. Upstream proxies serve a large number of users and have a better probability for finding a WWW content for the request. Hierarchical routing of the requests should be implemented for this function. Distributed caching requires distributed routing to be done between proxy servers, which work together as a distributed cache; load balancing, fault tolerance and scalability should be supported for this function. Active caching can work with different algorithms. For example [Bestavros & Cunha, 1996] and [Almeida et al. 1996] specify temporal locality of reference and geographical locality of reference. The former implies that recently accessed objects are likely to be accessed in the future, the latter, implies that an object accessed by a client is likely to be accessed again in the future by "nearby" clients. Thus it may be reasonable to pre-fetch the most popular and most recently accessed files among those which are in the cache and have expired.

Today, among the publicly and commercially available caching proxies, Microsoft Proxy Server 2.0 includes most of the described functions. We considered three different caching proxy software products: Microsoft Proxy Server, Netscape Proxy Server and Process Software Purveyor. We compared the differences in their performances, including cache response time for cached objects. Netscape Proxy performed significantly worse for small objects (up to 200 KBytes). Process Software Purveyor with a good average performance, showed

irregular service behavior. It was able to deliver about 40% of cached objects in very short time, while the others had an almost constant delay in service time. It can be related with a strategy for memory allocation or object retrieval from cache storage. In conclusion, Microsoft Proxy Server 2.0 was chosen for a basis for proxy caching service. It uses Cache Array Routing Protocol (CARP) [Microsoft 1997] for communication between different proxy servers.

2.2. CARP

CARP was designed as a potential replacement for the Internet Cache Protocol (ICP) [Wessels & Claffy, 1997a], [Wessels & Claffy, 1997b], the existing Internet standard for caching protocol, currently implemented in Harvest and Squid proxy cache packages. With CARP it became possible to create arrays of proxy servers, which work like a single virtual proxy server cache. According to [Microsoft 1997], two powerful benefits of CARP are:

- CARP uses hash-based routing to provide a deterministic request resolution path. Because of this there is none of the query messaging between proxy servers that is found with ICP, which creates a heavier congestion of queries as the number of servers increases.
- CARP eliminates the duplication of contents that otherwise occurs on an array of proxy servers. With an ICP network, an array of five proxy servers can rapidly evolve into essentially duplicate caches of the most frequently requested URLs. The hash-based routing of CARP keeps this from happening, allowing all five proxy servers to exist as a single logical cache. The result is a faster response to requests and a more efficient use of server resources.

ICP was designed with hierarchical cache in mind, and assumes that at any given single point in the hierarchy only one or two ICP servers as maximum are needed. CARP also supports hierarchical routing, but in addition it provides the possibility to build a large cache array (several machines acting as a single point cache) at one of the levels of hierarchy.

CARP makes it possible to plug additional servers into the cache array or subtract a server from the array without massive reconfigurations and without significant cache reassigning. Its cache-management features provide both load balancing and fault tolerance. CARP 1.0 was submitted as an Internet draft to the IETF.

3. Caching Proxy Configuration Policy

3.1. Caching Arrays Versus Stand-alone Caching Proxies

The benefits of CARP are mainly in its cache array management functions. But the key question is: “When does it become reasonable to configure cache array instead of using a stand-alone proxy caching server?” We made a set of measurements to find an answer.

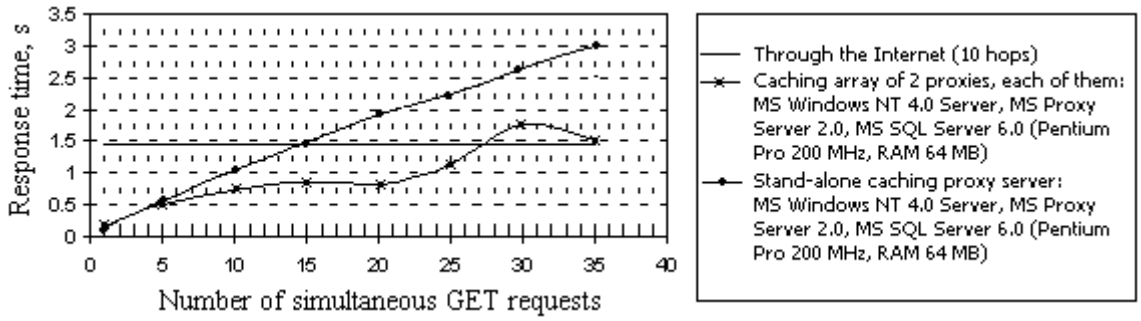

Figure 1: Average response time for the files of 100 KBytes, depending on number of simultaneous GET requests.

Different configurations of caches were tested under different loads, particularly the value of response time for the request served from the cache was measured for files of constant size under different server loads. [Fig. 1] represents the dependance of response time for the files of 100 KBytes on the number of simultaneous HTTP

GET requests. To simulate certain number of simultaneous GET requests, we used a correspondent to it number of processes, which were running in parallel on one client machine and were sending requests to the cache. Each of these processes was generating GET requests consequently (e.g. a process sends a request after a previous request was served from the cache). The client machine had the following configuration: Intel Pentium-II 266 MHz, RAM 64 MB, MS Windows NT 4.0 Server. In [Fig. 1] one curve corresponds to the stand-alone proxy server (Intel Pentium-Pro 200 MHz, RAM 64 MB, MS Windows NT 4.0 Server configured with MS Proxy Server 2.0 and MS SQL Server 6.0). Another curve corresponds to the cache array of two proxy servers (each of them Intel Pentium-Pro 200 MHz, RAM 64 MB, MS Windows NT 4.0 Server configured with MS Proxy Server 2.0 and MS SQL Server 6.0). And as a reference point the value of response time when the request was served from the Internet is presented; it was equal to 1.47 seconds.

In order to obtain a natural estimation of cache scalability from this data, we introduced a simple correspondence between the number of simultaneous HTTP GET requests and the number of cache users who browse the WWW at the same time. It was concluded from the statistics of our cache usage that an average user makes *R=4.433* GET requests per minute, we define *R* as the *user rate*. From our measurements we got the dependance of response time for the files of 100 KBytes on a number of simultaneous HTTP GET requests. If *m* is a number of simultaneous GET requests and *t(m)* is an average response time corresponding to *m* [Fig. 1], then *R*(m)=m/t(m)* is the *cache service rate*. Then the number of real cache users who browse the WWW at the same time can be approximately equal to the ratio of the *cache service rate* over the *user rate*.

$$N_m = \frac{R_m^*}{R} = \frac{m}{t_m R}$$

[Fig. 2] shows the same dependance as [Fig. 1] but on *N(m)*.

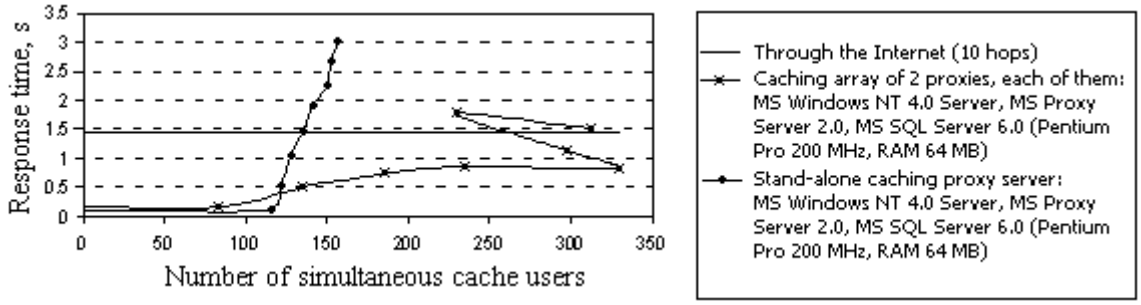

Figure 2: Average response time for the files of 100 KBytes, depending on number of simultaneous GET users.

From these results we can conclude that unless the number of simultaneous cache users is bigger than 100, users requests most probably will not influence each other's response time. Also we see that for the cache points where approximately up to 120 people use cache in the same time it is rational to build a caching service on only one stand-alone server, which could however be one of the levels in hierarchical tree of caching proxies. 120 users can be taken as an edge value. Until this value it is not reasonable to use CARP caching arrays since additional load related to computation of routing decision among array members is not compensated by the load, which users create for stand-alone configuration. Only for locations where more users accesses the WWW at the same time it is better to have arrays running with CARP, configured from two or more caching proxies. And the more servers that would form an array, the better the scalability one could get in this case. Since CARP assumes queryless caching, it avoids an increase of computing overhead while increasing the number of array members.

3.2. Logging to a Database

Logging to the MS SQL Server databases is possible option for keeping caching proxy logs. The decision on where and how to store proxy logs depends on many factors, mainly on the further actions, which are supposed to be done with logs. But in the case where it is decided to use logging to the MS SQL Server database, consideration should be given to the fact that the SQL Server by itself requires some system resources. According to our results, it is always preferable to keep the MS SQL Server database on a separate machine; otherwise the scalability of the caching proxy will be significantly lower.

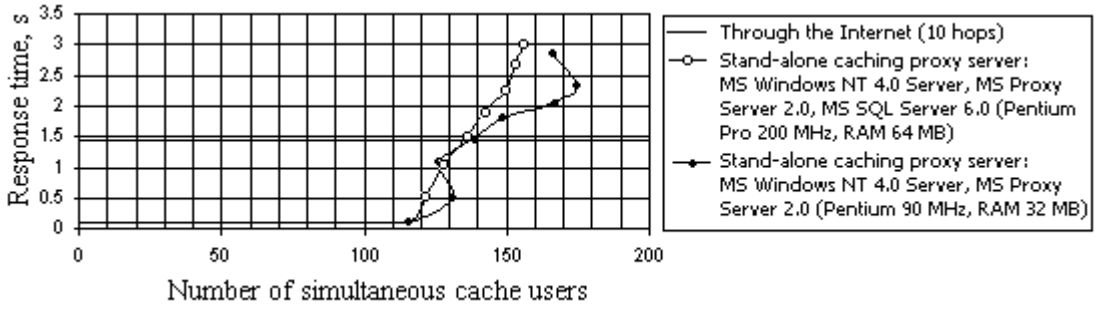

Figure 3: Average response time for the files of 100 KBytes, depending on number of simultaneous GET users; comparison for two different server configurations.

[Fig. 3] shows the comparison for two servers configured as stand-alone caching proxies. The first is Intel Pentium-Pro 200 MHz, RAM 64 MB, Windows NT 4.0 Server configured with MS Proxy Server 2.0 and MS SQL Server 6.0, which keeps its proxy logs in its SQL Server database. The second is Intel Pentium 90 MHz, RAM 32MB, Windows NT 4.0 Server configured with MS Proxy Server 2.0; it keeps its proxy logs in text file.

4. Caching Policy

4.1. Caching Versus Replication

In addition to traditional proxy caching, one could consider complete replication of the Internet WWW site to local storage for off-line browsing. This makes sense only for the sites that are not updated frequently, otherwise it would produce huge amounts of traffic; and taking into account that some parts of the context could be never requested by users does not sound reasonable [Baentsch et al. 1997]. The choice between caching and replication depends on several criteria. A set of experiments was performed in order to estimate their significance and to decide on the kind of service, that would be preferable to use.

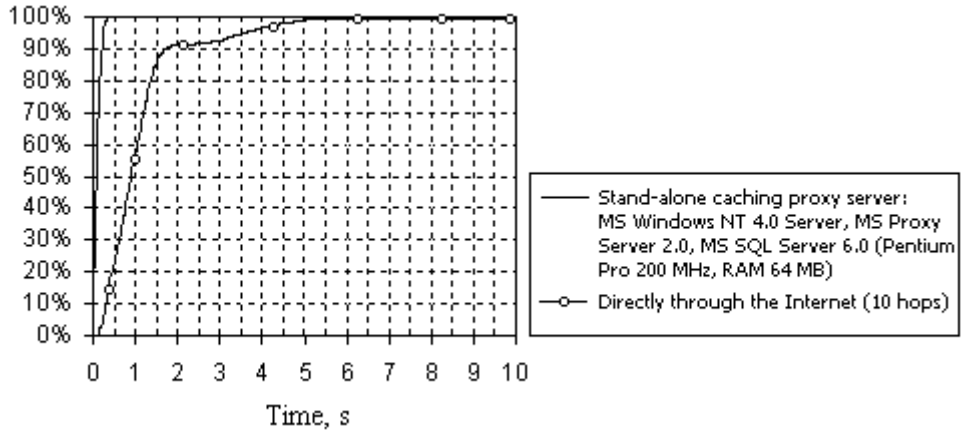

Figure 4: Download percentage for files up to 100KBytes.

Two sets of measurements were made for the two cases of download of WWW context. In the first case (I) files were downloaded from the Internet WWW server (10 hops away). In the second case (II) files were previously cached while making initial request through the proxy and were requested once again, in such a manner were downloaded from proxy cache via 10 Mbits/s LAN. [Fig. 4] and [Fig. 5] represent percentage of files downloaded with different response time values. [Fig. 6] shows the statistical data for the speed of the file download depending on the size of file. This statistics shows that about 2% of requests for relatively big files (of the size distributed randomly between 0 and 1E6 bytes) were not actually cached during initial request and

service speed was slowed down to the speed of the Internet. This corresponds to the lowest peaks on the [Fig. 6]. This can happen because connection to Internet server can be broken during download of big file due to the network congestion.

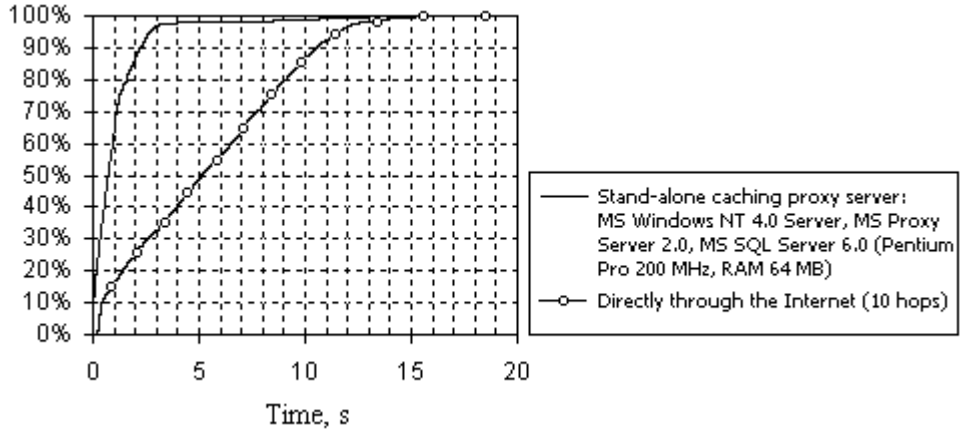

Figure 5: Download percentage for files up to 1MBytes.

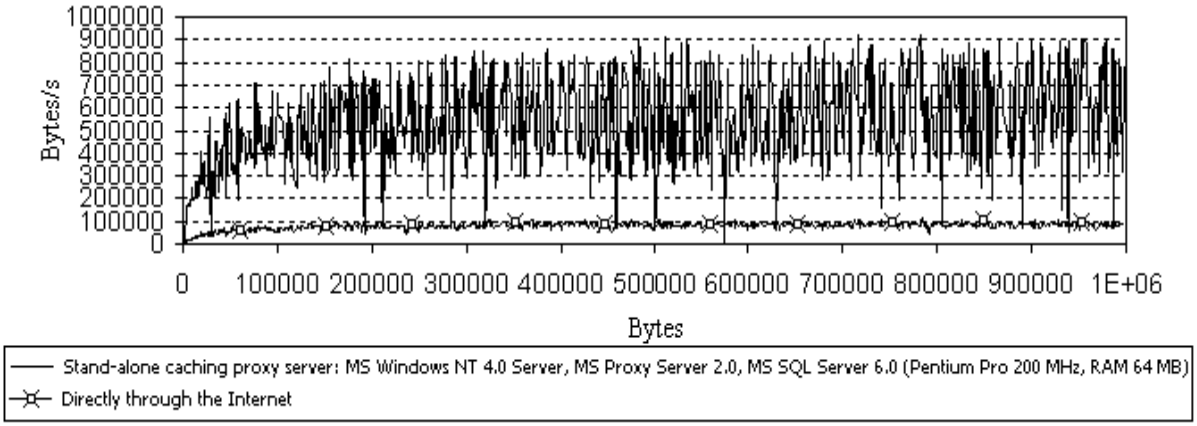

Figure 6: Speed of download depending on size of object.

Considering this statistics we can make difference between two kinds of WWW sites. The sites, which contain big amount of large files, form the first group. These are mainly sites containing a lot of graphics, such as, for example, Nicholas Roerich Museum (*http://www.roerich.org/*). In the second group there are "average" sites, most pages of which have total size of all objects of about 200 KBytes or less. The first group is primary target for replication, since broken connection, which can potentially occur while caching, could force user to wait more than 15 seconds until he/she gets just one of the objects on a page from the Internet. It is not reasonable to make replication of sites from the second group unless the site is accessible only through slow link (e.g. 10KBytes/s or less; for the reference WAN speeds are estimated [Microsoft 1997] typically 64Kbits/s – 1.5 Mbits/s). Therefore it is preferable to use caching for this group of sites. One shall note once again that all these suggestions apply only to the sites, which are not updated frequently, others shall not be replicated.

4.2. Updates Policy

As mentioned earlier, one of the two major advantages of caching proxies is the decrease of Internet traffic. This point became less valid after active caching was introduced. Some caching software products are able to automatically update files that are near to expiration or have already expired. The expiration time for an object normally should be provided by the WWW server, however this option is not used very often. Either caching

proxies take original expiration time from the WWW server or if it was not provided, a caching proxy may set this parameter for its cache according to the object's lifetime taken from the date of last modification of the object.

The proxy administrator can manage the amount of files updated and the frequency of updates. The main criterion, which should be considered here, is the number of proxy users. The value of geographical locality of reference [Bestavros & Cunha, 1996] becomes smaller and the number of objects becomes bigger while the number of users increases. Hence when the number of users increases it is reasonable to reduce at first the amount of files updated and afterwards consider the reduction in update frequency.

5. Future Research

Our conclusions could be useful recommendations for decisions in cache management. However, they may not be enough, in all cases to satisfy the users in regards to response time for HTTP requests. One of the interesting trends that merits examination is the power of Internet Agents that could enrich the WWW caching service. For example, the idea of "smart browsing", where according to the trace of previous user's requests, some probable variants for future requests are predicted and pre-fetched. This could significantly improve service, as users would probably not pay anymore than the price of the first click. With the current technology for WWW caching, the user always goes to the Internet if his/her URL has never been requested before.

6. Conclusions

According to our results, cache users do not influence each other's response time for HTTP requests unless there are more than 100 (approximately) people who use proxy cache in the same time. For small and medium size enterprises (approximately up to 120 web users) it is rational to build a caching service on only one stand-alone server, which could also be a level of hierarchical tree of caching proxies. For locations where more users access the WWW at the same time, e.g. corporate caches or ISP caches, we recommend using cache arrays running with CARP, configured from several caching proxies. When a cache array is used, the more servers form an array, the better the scalability it has. CARP caching in general is preferable to ICP.

Considering the possibility of keeping the MS Proxy Server logs in the MS SQL Server database, when WWW caching is the primary reason of the server, we recommend having the database server running on a different computer.

Site replication can be an alternative to WWW caching in the case where the site is not updated frequently and most pages of the site have a total size of all objects greater than 200 KBytes. Even then caching is still preferable, because statistically cached data were found to be 98% reliable in any scenario.

7. References

[Bestavros and Cunha, 1996] Bestavros, A., & Cunha, C. (1996). Server-initiated Document Dissemination for the WWW. *IEEE Data Engineering Bulletin*, 19(3): 3-11, September 1996.

[Almeida et al. 1996] Almeida, V., Bestavros, A., Crovella, M., de Oliveira, A. (1996). Characterizing Reference Locality in the WWW, *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Dec. 1996.

[Microsoft 1997] Microsoft Corp., *Microsoft Cache Array Routing Protocol*. White paper.

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

[Wessels and Claffy, 1997b] Wessels, D., Claffy, K. (1997). Application of Internet Cache Protocol (ICP), version 2. *RFC 2187*, September 1997.

[Baentsch et al. 1997] Baentsch, M., Baum, L., Molter, G., Rothkugel, S., Sturm, P. (1997) Caching and Replication in the World Wide Web or a European Perspective on Connectivity, *3rd CaberNet Plenary Workshop*, IRISA, Campus de Beaulieu, Rennes, France, April 1997.