World-Wide Web Caching -- The Application level view of the Internet

GeneSys
CS Department, University of Kaiserslautern,
P.O. Box 3049, D-67653 Kaiserslautern, Germany
Email: genesys@informatik.uni-kl.de

The Internet has fallen prey to its most successful service, the World-Wide Web. The networks do not keep up with the demands incurred by the huge amount of Web surfers. Thus, it takes longer and longer to obtain the information one wants to access via the World-Wide Web. Many solutions to the problem of network congestion have been developed in distributed systems research in general and distributed file and database systems in particular. The introduction of caching and replication strategies has proven to help in many situations and therefore these techniques are also applied to the WWW. Although most problems and associated solutions are known, some circumstances are different with the Web, forcing the adaptation of known strategies. This paper gives an overview about these differences and about currently deployed, developed, and evaluated solutions.

Introduction

Applications created on top of the basic Internet protocol infrastructure caused this network of networks to become as successful as it is today. The electronic mail service was arguably the first truly online service any user could benefit from. Due to the increasing processing capabilities of the single machines connected to the Internet, new and more demanding services have been developed. Multimedia mail, computer/video conferencing, and last but not least very easy to use graphical front-ends to the wealth of information accessible via the World-Wide Web stressed the Internet to its limits. Applications circumventing such limitations by greedily adapting to available bandwidth further worsen the problem. Therefore, new enhancements to the infrastructure of the Internet were driven by these new realities arising from its own success. The development of a next generation IP protocol, IPv6, should only serve as one example for the tremendous, if only indirect influence of applications on general Internet protocols and services.

Certain problems, however, cannot be dealt with on the rather low level of the Internet infrastructure consisting of backbone communications links, routers, and name servers, e.g. The problems of interest to this article are the reduction of user-perceived document access latency and the issues of network usage optimization for the World-Wide Web. It is shown that both goals are at least partially contradictory to one another and can therefore not be dealt with by Internet protocols beneath the WWW itself. Nevertheless, some techniques can be applied at application level to solve both of them, namely caching and replication of documents.

This survey sets out with a brief general overview about caching mechanisms developed for different components in computer and communications subsystems, putting a strong emphasis on caching strategies in the Web. The technical foundations of the current caching infrastructure in the Web are introduced before several projects are high-

lighted that aim at providing improved WWW access. The central part of this survey consists of a collection of experience reports about the effectiveness of caching in the different parts of this Internet service. Lessons learned and results to be applied to the core mechanisms of the Internet are derived. In order to complete this survey with a glance into the future, concepts currently developed or under discussion are also included. In this respect, sections on active data replication including naming and fault tolerance issues, as well as security and accounting problems relevant to caching in the World-Wide Web conclude this paper.

A short history on caching

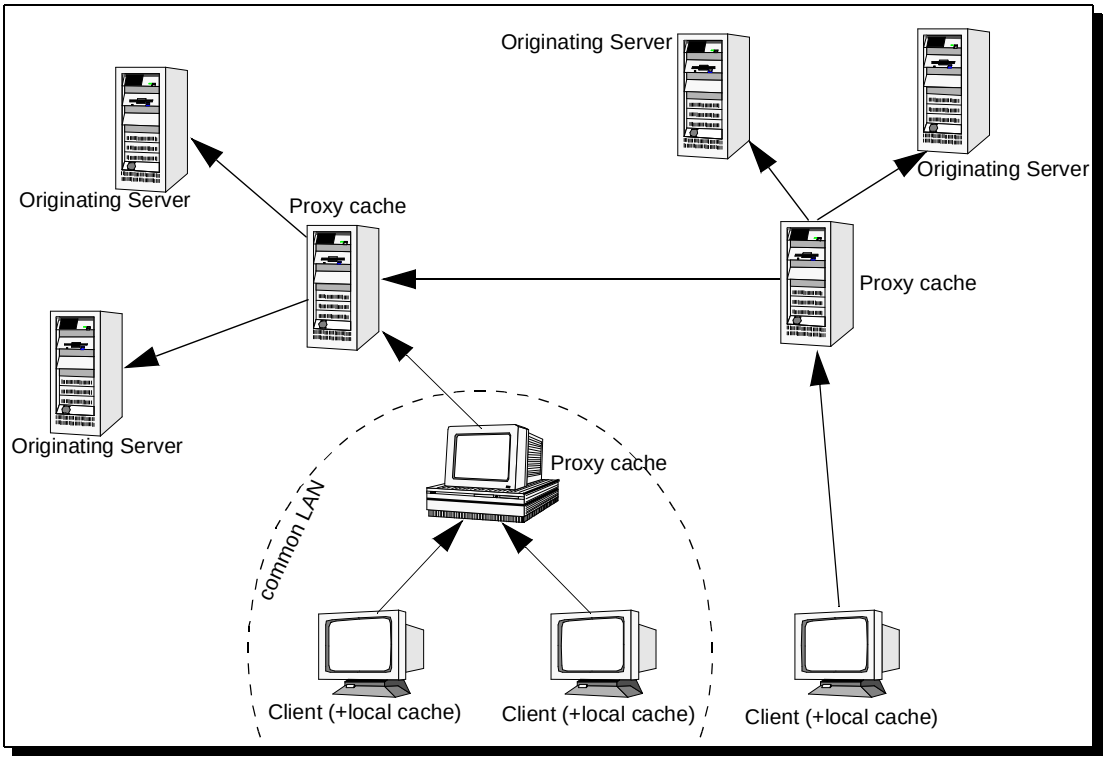
Caching is a technique generally aimed at bringing parts of an overall data set closer to its processing site. It is applied for example in memory hierarchies where relatively small on-processor caches yield data access times a magnitude lower than memory-chip access times. The same principle, although on another size and latency scale, also applies to the filesystem, where frequently used documents are kept in main memory instead of being retrieved slowly from disk. The Internet expands this memory hierarchy one further step outward from the processor. In principle, by keeping frequently requested documents close to where they are needed, significant reductions in access latency and bandwidth usage can be achieved.

This observation is far from being new. In distributed filesystems like AFS, this property is exploited in order to build a system that can scale to thousands of interconnected workstations sharing the same files. However, in the course of development of the World-Wide Web significant differences to distributed filesystems surfaced, causing the whole development of caching and replication strategies to be repeated, though in a much more rapid way, building on the experiences with previously built caching hierarchies. Among the observed differences, notoriously high latencies, the extremely large user community and the absence of a clear reference locality contributing to the success of hitherto known caching strategies clearly stand out as reasons why new or adapted techniques were needed.

Caching on the Web began with the introduction of *client-local* document stores either implemented as host-local disk backup store or as in-memory caches. The disadvantage of employing this approach alone became obvious as soon as too many different documents appeared on the Web, rendering the formerly known notion of locality set obsolete. Although client-centric caches can achieve surprisingly high hit rates as shown later in this survey, the nature of the Web calls for caches shared by many people and/or people with similar interests. This observation, together with a need for controllable access to the wealth of information on the Internet via firewalls, led to the development of *caching proxies*. Proxies in this context are programs accepting document retrieval requests from a client, thus acting as servers towards these clients. In turn, these proxies obtain the requested documents from the originating site as indicated in the address of a document, its URL. Thus, it became administratively possible to control the access from within an intranet that is otherwise disconnected from the Internet for reasons of security. The more interesting feature of proxies in this context is their ideal position to remember frequently requested documents and to keep them ready for subsequent requests by possibly different users of the proxy, effectively creating a WWW document cache. The first widely available software providing this functionality was the CERN WWW server as made publicly available in 1993, although others also created proprietary solutions at about the same time [X, Gla94].

With the rising use of frequently changing page contents, the problem of document accuracy, resp. staleness came up. Although the reference implementations of caching proxies already provided the possibility to specify the time documents should expire from caches, virtually no WWW server makes use of this feature. Therefore, caches had to come up with heuristics to determine the staleness of stored documents in combination with occasional polling of the originating site. This in turn led to an ever growing tendency of document providers to deliberately keep caches from storing their documents, i.e., defeating their original purpose in order to be able to always deliver the most current document to the client.

In the recent past, software aimed at reaping the benefits of several caches in hierarchies of caches has been developed. These hierarchies range from server-site caches over intermediate proxies down to client site caches effectively enlarging the overall cache size. Topics like load-balancing, dealing with communications link and host failures, as well as access-anticipating document mirroring are currently being addressed.



Technical foundations

The World-Wide Web's Hypertext Transfer Protocol (HTTP) as an application-level protocol on top of TCP already contains protocol fields aimed at supporting caching transactions between originating servers and intermediate as well as client side caches. In the following section, an overview about these protocol components as specified in HTTP version 1.1 is given [X, FG+96] although a strong focus is laid on the currently available HTTP 1.0 protocol fields.

The most important field, the Expires header is sent by virtually no server. It indicates the time after which a respective document may no longer be served from a cache. Ignoring this field leads to a situation in which caches have to apply time-to-live (TTL) heuristics to determine when some documents are stale and are thus no longer eligible for staying in the cache. When creating such heuristics, most cache developers use the Last-Modified header mandatory in any answer from an HTTP server. This entry indi-

cates the last time the respective document has been changed on the originating server. It is specified like all time fields in Greenwich Mean Time (GMT). The heuristics usually base on the experience that documents that were not modified recently will not be changed in the near future, whereas recently changed documents should not be kept in the cache for too long since they will probably be modified rather swiftly. In addition to this, some caching proxies also permit the administrator to specify sets of URLs or document types for which to apply special rules, e.g. never to cache URLs containing the string "cgi-bin" since this often indicates dynamically created data. As soon as a cache decided that some given document has to be considered stale, it has two choices. Either, the document can be delivered nonetheless, or -as is usually done- the document accuracy is checked back at the originating server. The first solution is only sensible, if the client has otherwise no possibility to connect to the network or if some new version of the respective document can be sent later on to the client as soon as it arrives over the network as proposed in [X, DiP96]. In order to make the second option network efficient, the If-Modified-Since (IMS) header has been introduced into HTTP. If a document's accuracy is unknown and has to be newly determined, a caching proxy can send a document validation request to the originating site. The time indicated in the IMS header designates the last time a cache obtained a new copy from the originating site. Servers receiving this request have two options. They can either answer with the most up to date version of the requested document, if it has changed since the time indicated in the IMS field, or they may simply return HTTP answer code 304 indicating no modification to the original data. The second option saves the bandwidth of a new transmission of the possibly large chunk of data addressed in the request. In either case, a new LastModified header is sent to the cache, which is used again to calculate the next probable expiration time of this document. The last HTTP header influencing the time-to-life heuristics employed by caches is the Date field. HTTP 1.1 Web servers are obliged to send this information indicating the time the document has been sent out by the originating server. This field can be used by caches to determine the absolutely possible staleness of a document by computing the difference between the current time and the date indicated in the Date field. This difference can also be explicitly sent in an Age protocol field to be produced in response to document requests by caches conforming to HTTP 1.1. In the opposite direction, the Pragma header is arguably the most important protocol field sent by a client to a server via any cache. One option, no-cache, forces all intermediate caches to only deliver up-to-date information, effectively renewing the requested document in all caches between client and server. It is sent for example, if a user pressed the Reload button in a WWW browser to request the most current version of some document. For discussions of other WWW caching heuristics see [X, AS+94], [X, BoH96], or [GwS96].

Some new protocol headers concerning caching have been introduced in version 1.1 of HTTP, which is not yet widely supported. Since they provide a significant improvement over the basic mechanisms described above, the most important of them are listed here for completeness. The Cache-Control: max-age header is a new option to specify the maximum amount of time a document may be kept in caches as permitted by the originating server. This option supersedes the Expires header. The Cache-control: max-age header may also be sent by clients in order to explicitly and flexibly specify the degree of staleness acceptable to the user. This option supersedes the more inflexible no-cache Pragma directive that could only be used to flush caches unconditionally. Entity tags realizing opaque service-level hints about the accuracy of cached vs. original data form other new header fields in HTTP 1.1. The rationale behind them is that

document creators are the only people able to specify that their documents changed in a semantically significant way by issuing new entity tag entries.

Caching Infrastructures

There have been several projects aimed at building single software packages or complete infrastructures supporting caching in the WWW. We are presenting three typical, prominent, and well-documented systems that differ widely in their approach.

CERN httpd

The first widely available and still most popular WWW server has been developed at the birthplace of the World-Wide Web, the CERN laboratories in Geneva [LuA94]. The original server has been enhanced by proxy caching features in early 1994. This software gained widespread acceptance and served as a reference implementation to many caching proxies developed later on. Technically however, it is inadequate specifically when large numbers of requests have to be handled. This is mostly due to its simple mapping of URLs to cache filesystem names resulting in expensive lookup operations on every request. Moreover, a new, full-fledged UNIX process is created for every request, leading to huge memory and CPU loads on high access rates. The CERN proxy was also the first one introducing the continuous loading of pages even after a user interrupted the download, e.g. because of a long delay, a feature considered by some to be a flaw. On the one hand side, this makes sense given the (high) probability that the originally requesting client will try again sometimes later. On the other hand, however, this continued loading binds resources urgently needed on the proxy, especially during the access-wise most busy times of day. Finally, the conceptual flaw that caused this software to be of no use in cooperative caching schemes is discussed below. CERN caches can only be configured to use one further cache higher up in a thus strict hierarchical tree of caches. Moreover, it could not gracefully deal with the failure of such parent caches by circumventing them for example, but ceased to deliver documents under such circumstances completely.

HENSA

One of the first and best documented approaches to building a coherent large caching infrastructure started in late 1993 in the University of Kent at HENSA Unix [Smi96]. The goal was to provide an efficient national cache infrastructure for the UK. After initial problems with inadequate hard- and software, one single, centrally administered cache has been created that handles currently over 1 million cache requests per day. Many UK institutions use HENSA as a cache behind their own caching proxies. Technically, HENSA used Lagoon, a rather immature proxy, for a short time before switching to the CERN proxy software. As soon as many people started using the HENSA caching service, the limitations of the software as outlined above became apparent, and finally a commercial caching proxy from Netscape Communications was installed on a set of six machines sharing their load. This is achieved by making use of several DNS resource record entries [X, Moc87] for the one Internet name of the HENSA caching proxy that are returned round robin to clients resolving the cache’s name to IP addresses.

This approach of a centrally administered caching service subsequently faced several barriers inherent to a non-distributed approach. Each of them had to be dealt with by migrating to new software or new hardware to keep up with demand. Whether the cur-

rent configuration suffices for future employment with ever more users remains to be seen.

Harvest

Probably the most sophisticated software cashing in on the experiences from previous research in caching resulted from the Harvest information discovery project [CD+96]. It sharply contrasts in concept from the HENSA approach by enabling administrators to span a tree of cooperating caches over wide distances. Harvest caches can be arranged in parent or sibling relationships with each cache contributing to the overall data set in a generally autonomous way. Whenever a caching proxy cannot serve a requested document from its local cache, it sends messages to its parents and siblings querying whether they hold the document. These messages are sent in parallel, and the first peer returning a positive acknowledgment is asked to deliver the document in question. This technical opportunity to link together caching servers to obtain a better general service resulted in a network of more than 1000 registered, and cooperating caching proxies all over the world [X, NLA96]. This again hierarchically delegating registry aims at overcoming the administrative problems of independently operating caching proxies, a feat sound in its own right regardless of technology employed.

The technical improvements implemented in the Harvest cache aim at optimizing nearly any component on the critical path from user request to delivery of the data. Positive as well as negative DNS queries for IP name to address resolutions are cached, and the most recently requested documents are kept in main memory. Moreover, one single UNIX process handles all requests via non-blocking I/O system calls, obviating the need for a distinct process per request. Additionally, disk management is optimized in so far as cached data is evenly distributed over the drives assigned to the proxy server. The disk metadata is completely kept in main memory obviating the need for expensive file system lookups on every request when the availability of some requested document has to be determined. The combination of all these improvements yield a program able to serve documents roughly two orders of magnitude faster than the CERN caching proxy while causing a much lower system load.

Additionally, this caching proxy took caching one step further towards the original server by providing a mode in which it can be run as an accelerator for origin servers. It takes on the role of a normal WWW server albeit without access to a filesystem, only serving recently requested, popular documents from its in-memory cache. As soon as a cache miss is encountered, the accelerator resorts to the original server, which resides on the same host, obtains the missing document, and stores it again in memory, possibly after flushing least recently used data.

Another highly efficient, modern, and particularly easy to administrate cache has been developed by Netscape Communications. Since it is usually only available with a rather hefty price tag, it did not gain the widespread acceptance of the freely available software of CERN or Harvest.

Analysis

Even though the Harvest cache [X, Har95], and its public domain descendant Squid [X, Wes96], are clearly superior due to their features, the original CERN proxy is still in widespread use throughout the Internet. In order to substantiate this proposition, we evaluated our university’s main WWW server’s logs in order to find out how many requests are served via proxies and which proxies are in use. This is possible because

proxies commonly add identification information to the one sent by clients to servers, thus making their presence visible. Thus, originating servers can react in a special way for some cache/client combinations.

The first lines in the table below show the percentage of accesses to the main WWW server from outside our university made via particular caching proxy software. In the second to last line the overall percentage of requests made via any proxy in relation to all accesses is shown. The last line indicates the amount of requests that passed more than one cache on their way to this server. This is also the reason why the sum of the lines for CERN, Harvest, Squid and other caches may result in more than 100%.

Table 1. Distribution of caching proxy accesses (in percent)

1996	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
CERN	88.05	83.47	77.49	74.64	70.14	64.64	68.50	57.89
Harvest	12.44	18.00	24.81	26.49	21.96	18.35	11.45	13.08
Squid	0.00	0.00	0.00	0.52	10.65	19.88	22.74	31.83
Other	0.35	0.35	0.50	0.48	1.19	1.23	0.94	0.68
Overall	*29.28*	*25.63*	*21.04*	*19.41*	*24.94*	*25.40*	*25.71*	*23.77*
Multi	0.84	1.83	2.81	2.14	3.94	4.10	3.63	3.47

From these data, one can conclude that the use of proxy caching (in addition to standard client caching) is a well-accepted approach and that administrators react rather swiftly on the advent of advanced technology.

Experiences

This section gives an overview about the results various research groups as well as administrators of cache sites gained when evaluating their caches' performance. We back these informations gained by others through a comprehensive evaluation of the logs of our University's as well as our research group's main caching proxies that we instrumented especially for this purpose. In order to fully clarify the environment these servers run in, some technical details may be in order. The hosts (Sparc ELC, resp. DEC Alpha) are connected via Ethernet LAN and a 34 MBit/sec. link to a german network backbone. Both servers run particularly instrumented versions of the CERN proxy server with 600 MB disk space seeing daily request rates in the order of 20000, resp. 500 for the group local cache. The main proxy served 48.9 GBytes in 5.8 million requests during the 10 months surveyed.

Hit rates and latencies

One may wonder whether caching is worthwhile at all given the immense wealth of information accessible via the WWW. In order to stress this point we determined the number and size of documents making up the most popular pages obtained via our caches extending results reported by other groups [X, Cla94]. In the table below we list the percentage of data in the first line that account for the amount of byte hits as shown in the second line, and what percentage of the cache's (disk) space they require. For

example, 0.25% of all cached pages are responsible for nearly 6% of all cache byte hits while retaining only 0.1% of the overall disk space.

Table 2. Locality as seen in 1996 (in percent averaged over 8 months)

	0.25	**0.60**	**2.75**	**6.25**	**14.00**
Accounting for	5.96	9.22	20.94	30.20	42.35
Occupied Space	0.09	0.23	1.05	2.58	6.94

These numbers exemplify the point that something like reference locality does indeed exist in the World-Wide Web. This effect probably stems from word-of-mouth propagation about "hot sites", sites producing continuously interesting contents like magazines, or certain pages very easily accessible via particular browser's navigational buttons. For example, the Internet search site document as accessible via the Net Search Directory button in the omnipresent Netscape Navigator is one of the most popular pages throughout the months surveyed.

As the ultimate measure of a proxy cache's efficiency, we also measured the absolute cache hit rate as well as the byte hit rate during the year 1996. The numbers obtained back previous findings by other groups that intermediate proxies typically obtain hit rates ranging from 8% to 45% depending on the user community and whether (LAN-) locally served documents are counted or not. In addition to these numbers, statistics provided by the National Laboratory for Applied Network Research's country-wide caching federation [X, NLA96] based on the Harvest software point in the same direction. Also included in the table below are simulations on theoretically possible ideal cache hit rates separately for each month disregarding cache size and document staleness, i.e., documents accessed once are supposed to never change and always to be delivered via the cache. The last two rows show hit rate and byte hit rate as would have been obtained by the above mentioned ideal cache warmed by accesses starting in November of 1995 (not shown here) and run through June 1996. Months July and August could not be simulated due to memory allocation problems on the machines we used for the experiment.

Table 3. Hit rates as seen during the year 1996 (in percent)

	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**
Hit Rate	19.28	20.05	21.39	20.71	20.20	20.98	22.36	26.56
Byte Hit Rate	21.12	19.71	19.28	17.68	17.10	17.24	18.38	20.46
Per month Ideal Hit Rate (IHR)	46.37	44.87	43.18	44.96	45.77	44.46	46.95	42.11
Per month Ideal Byte Hit Rate (IBHR)	41.36	39.56	36.62	37.00	35.84	35.20	39.02	32.67
Cumulative IHR	53.58	54.36	54.67	55.30	55.92	56.23	N/A	N/A
Cumulative IBHR	48.72	49.61	49.91	50.21	50.30	50.34	N/A	N/A

The major result from these simulations is the statement that regardless of the quality of caching strategy and cache size, 50% appear to be an upper bound to optimal byte

hit rates seen on a reasonably well loaded proxy. These findings further agree with hit rates reported earlier on small caches [X, AS+94], and byte hit rates in [BC+95] stating that cache hit rates appear to drastically level off at a rather low cache size of 100 MB. This is a result not so surprising if one takes into consideration the (non)locality as observed above as well as the fact that proxy caches only get those accesses not already served by earlier client caches. On the down side, the table above shows the rather small corridor in which improvements on caching strategies for the WWW can take place. This in turn raises the question whether further, more sophisticated caching strategies are worthwhile developing, or whether research should concentrate more on directly user visible and influenceable strategies.

There is, however, one big difference to reports like [X, MuH92] which argue that cache hierarchies for file systems are merely latency *creation* mechanisms. It is the environment in which the WWW operates that is totally different from the LAN environments as considered in that work. In a LAN, one can count on relatively low latencies and rather high bandwidth causing even slight delays in intermediate caches to produce transmission degradations perceptible to the end-user. In the WAN environment of the Internet, latencies induced by caches are basically negligible. In order to underline this point of view, some groups conducted experiments with a variable number of sequentially arranged intermediate caching proxies accessed by several users. They found that a chain of up to 3 transparently accessed intermediate proxies is commonly not noticed by users. Only if 4 or more caches lie between client and originating server, users start getting annoyed by additional latency introduced by the proxies. We support these findings originally obtained from surveying individual users by an objective experiment. None, 1, 2, or 3 proxies were used to access two sites, one of which is in WAN distance (*WAN measurements* in the table below) from our proxy hierarchy whereas the other is on the same campus (*LAN measurements*). The different stages were running the CERN proxy, resp. Squid software on Sparc 10, Sparc ELC, and DEC Alpha in different configurations. Three runs per configuration were made, the average time for the retrieval of 284 images (in milliseconds per document) are shown in the table below. One caveat however is in order when judging the wide area measurements. The variance observed between the single test runs is very high, which is why basically the absolute loading durations as measured for the LAN case are much more relevant than the WAN times.

Table 4. Absolute latencies introduced by the use of additional proxies (msec/access)

	0 / direct access	**1 Squid proxy**	**1 CERN proxy**	**2 proxies**	**3 proxies**
LAN measurements					
Duration (cold caches)	240.42	274.10	335.65	359.15	602.78
Degradation	-	14.0%	39.6%	49.4%	150.7%
Duration (warm caches)	240.42	150.99	181.18	240.41	334.49
Speedup	-	37.2%	24.4%	0.0%	-39.1%
WAN measurements					
Duration (cold caches)	2443.66	2616.20	2985.12	2922.54	4048.50
Degradation	-	7.06%	22.16%	19.59%	65.67%
Duration (warm caches)	2443.66	91.55	123.23	140.85	147.89
Speedup	-	2669.21%	1983.00	1734.94%	1652.35%

These numbers not only stress the point that a hierarchy of up to three chained proxies does not add really noticeable delay to the retrieval of a single document, but the comparison of the LAN and WAN measurements make clear that caching proxies should not keep documents regularly accessible at LAN transfer rates. Otherwise a proxy hierarchy only functions as a latency inducer as pointed out in [X, MuH92].

As already briefly mentioned in the section on the history of caching in the World-Wide Web, there are several places where caches can be installed. The first one is the client side. In [BC+95] the authors report on their experiences with instrumented Mosaic clients reporting cache hit rates and user behavior. The basic finding was that surprisingly high hit efficiency (77%) although at a much lower byte hit rate (33%) can be achieved. Based on these logs, simulations were performed in order to determine the benefit of caching proxies shared by several users on the same host and by several users on the same network. The results were quite interesting in that they showed only slightly higher overall cache hit rates albeit significantly reduced host loads for shared caches as opposed to client-local ones. Our findings obtained from evaluation of real-world traces from our instrumented caching proxies support these results. When studying reports on proxy cache hit rates, one must always keep in mind that they always are only third or even higher level caches in a document access hierarchy beginning in the WWW browser's main memory. Therefore, hit rates as reported in table 3 should be not surprising and should further not be mistaken with client cache hit rates usually somewhat higher due to a single user's locality behavior.

One general remark on the comparison of hit rates as reported in different papers is clearly necessary. Some of the pure hit rate percentages reported can clearly be misleading since the effective byte hit rate is usually lower. The latter, however, is the measure that counts as long as no generally agreed-upon measure of cache efficacy is available. The reasons for this are that network bandwidth savings have to be measured in bytes and that user perceived latency clearly depends on available bandwidth again measured in bytes (per second).

As far as latency reduction is concerned, we could confirm the statement issued by the providers of the HENSA UK national cache [X, Hensa] by our own measurements that on average 25 seconds per document retrieved via one proxy cache alone are being saved. This number averages over cache hits and misses alike giving a fairly simple measure for the end user advantages reaped by proxy caching.

In conclusion, the hit rates that can be observed are rather low when looking at the single stages in isolation, but in combination a surprisingly good payoff for multi-level caching becomes obvious. These findings are clearly an incentive for users to take advantage of cache hierarchies and for service providers to participate in such cache federations.

User education

One of the surprisingly difficult problems experienced by most cache service providers however is the issue of how to educate a user community to make use of caches besides their own client-local caches which are in most cases automatically managed. No problem arises if users are administratively forced to use some given (firewall) proxy due to security or connectivity constraints. In an environment in which users do not have the technical background necessary to fully grasp the concept of caching, the very use of it has to be carefully explained and side effects like stale data as well as ways to deal with them have to be pointed out explicitly. It turns out that complete transparency of cach-

ing as achieved in file system, or memory system caching is not possible on the Web. The reasons for this are on the one hand the user perceivable lack of cache coherence and on the other hand the large number of different server, proxy, and client implementations with often radically differing goals. Cache administrators for example always strive to optimize network utilization, while users want to obtain the most up-to-date version of any data.

Lessons learned

One interesting prediction stated in [GwS96] concerns the rising number of dynamically generated documents. By monitoring the documents retrieved via our university's proxy cache, we were able to find out that this statement seems to need re-evaluation. The rate of documents resulting from POST, querying GET commands, or otherwise non-cachable data hovered around 6% and never raised above 8.5% during the whole year we monitored the traffic on the proxy. This observation is a fortunate message for any caching system designer. A reason for the low rate of dynamically generated documents may be that it turns out to be too expensive to be performed additionally on an already well-loaded Web server machine.

One further good message for cache designers is the observation stated in [X, Be95b] that the most popular documents on a WWW server change less often (0.5-2.0% change probability per day) than other data on the same server. This result lead for example to promising simulations of adaptive, bandwidth-saving cache consistency protocols [GwS96]. In [Wes95] the mean bandwidth seen when a cache miss occurs is reported to be 22kbps, whereas we measured one of merely 3.2kbps. This is mostly due to the small bandwidth of transatlantic links between europe and america in addition to increased traffic between the measurements made by Wessels and ours. This observation again underlines the need for good caching algorithms especially when considering another difference between the measurements of [Wes95] and the ones we did. Wessels reports a mean time between requests for the same document in the cache to be 2.5 hours. In our observation this time appears to be 42 hours, clearly showing the growth of the Web in terms of accessible documents, a fact to be taken into consideration when designing caching proxy strategies.

Another topic of interest to us is the amount of requests and the byte transfer rate induced by the various search engine crawlers or robots automatically retrieving and indexing WWW documents. Therefore, we counted the accesses by these programs and it turned out that the robot-induced data retrieval rate seen on our server increased from 1.72% in November 1995 to 6.25% of all bytes retrieved in August 1996. These numbers point towards a problem that should be considered although it is currently not yet sharply pronounced. Caching of document meta information suitable for WWW document crawlers for example might be a beneficial approach to the servers thus relieved of unnecessary accesses. This approach also benefits the various search engines in that it saves all of them the compute overhead currently necessary for evaluating document changes.

As a final metric presented in this survey that might influence caching decisions, we are looking into the numbers of document transfer interrupts by users as exercised via a WWW browser's Stop button. It turns out, that 11% of all requests made via our proxy are cancelled because of a user's impatience. We noticed that some documents were very regularly tried to load, indicating a very high interest of users in the particular documents. The effectiveness, resp. user acceptance of a caching strategy built on this

observation might be an interesting feat. Documents requested repeatedly but not loaded due to network congestions, can be obtained in a preferential manner, e.g. more persistently or automatically during off hours.

In order to determine whether cache validation messages are relatively seldom as compared to standard cache transactions, we measured the occurrences of the HTTP response 304 indicating an originating server's response that nothing changed in the document in question. To this end, we again used the logs of our university proxy cache and determined a surprisingly high rate of about 10%. If this finding is representative, a special kind of caching proposed by several researchers becomes interesting: Active cache invalidation [X, Wes95], prefetching [X, WaC96], mirroring, push-caching [X, GwS95], or replication as it has been dubbed by different groups.

Replication

In order to reduce the relatively high amount of client-driven cache validation messages as proven to exist thus reducing the user-perceived latency, it seems to be sensible to replicate certain frequently requested data, i.e., to keep cached data consistent with the original documents. This approach also parallels one of the final steps in the development of distributed file systems. Checking back with originating sites by caching proxies proved to be a worse strategy than a reliance on originating servers to keep replica sites informed about document changes [X, HK+88]. Document replication also offers further advantages. For example, documents that are assumed to be highly popular, e.g. new software releases, can be sent to replica sites in advance. Thus, the initial faulting of popular data into caches can be anticipated and done outside the critical path of document retrieval initiated by the single user. A further advantage of replica servers consists in the provision of fault tolerant services. As soon as sites are known to provide the exactly same data, they can be accessed, if the previously best replica or originating site becomes unreachable or unbearably slow, an event very easily detected by proxies further down in the hierarchy.

It has been pointed out that hierarchies of replica sites coexisting with caches promise the highest degree of scalability and manageability for filesystems [X, BlA92] as well as the WWW [BMS96]. The implementation of such a system proved to be rather simple, since it is basically just a special case of caching. Documents are no longer automatically replaced but have to be handled by a mechanism outside the standard caching strategies. The other fortunate factor is the virtual absence of modifying accesses to documents. The by far biggest portion of traffic on the WWW consists of cachable, read-only data (94% in the measurements of data access via our university's caching proxy). As soon as dynamic documents, i.e., pages changing per usage are accessed, one may fall back to the HTTP specification which explicitly states that caches have to write through all requests with dynamic access methods other than the read-only HTTP methods GET and HEAD to the originating servers. If that does not meet the service providers' needs, complete services have to be replicated which is feasible in most situations like search engines creating answers out of a mostly unchanging database of known documents.

In order to point out the advantages of replication in a numerically and graphically graspable way, we instrumented our proxy with another probe, calculating the average bandwidth for document transfers between the proxy and remote sites. The most clear one is shown in figure 2. It indicates the application-level bandwidth measured during the course of a normal workday averaged over the months February, March, and April

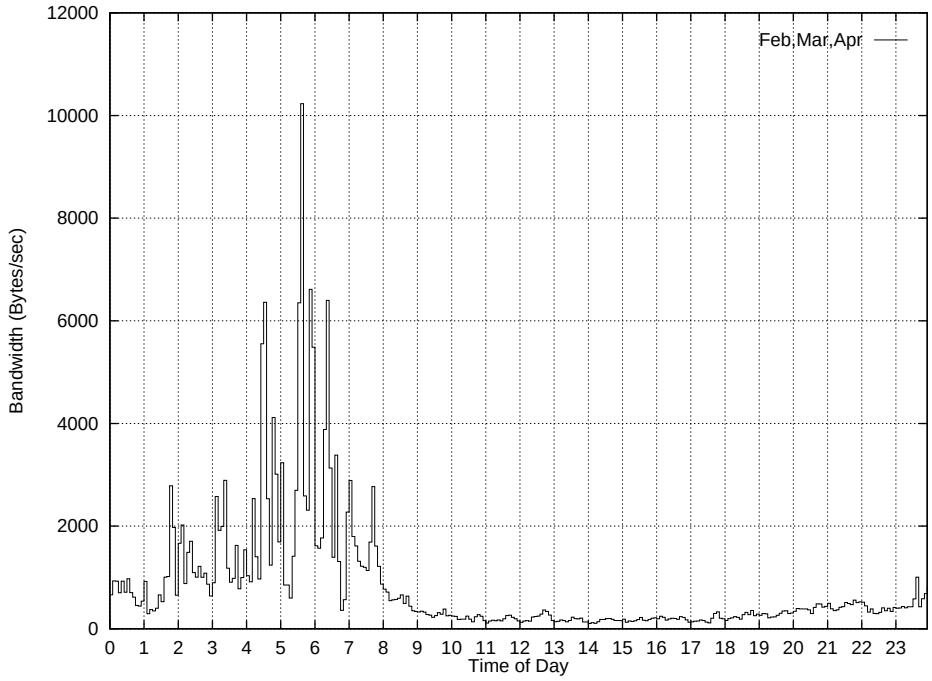


Figure 2: Bandwidth as seen between www.uni-kl.de and *.com hosts

1996. This graph shows vividly that order of magnitude bandwidth deviations exist between day and nighttime between our measurement point, the university proxy and hosts in the domain .com. Replication schemes can benefit ideally from such knowledge by delaying update operations to the off-hours around 6 o'clock in europe. Although WWW caching strategies based on these trends have been reported for example in [X, BoH96], they concentrate mainly on improving cache hit rates. Other groups concentrate on trading bandwidth for latency, for example in [X, Be95a] theoretical results based on small server traces predict 23% latency reduction for 5% increase in wasted bandwidth by sending not (yet) requested documents preemptively towards clients. However, conclusive results on optimal byte hit rates, server utilization, and user-perceived latency in combined caching and replication systems under different usage scenarios are yet to be developed for the World-Wide Web.

The naming problem

The biggest obstacle to the adoption of replication techniques is the lack of a location transparent addressing scheme for the World-Wide Web. Since URLs are closely linked to the DNS naming model (WWW addresses contain internet addresses), the problem is closely linked to Internet name resolution. Certain strategies exist that return different IP addresses if asked for a name resolution. This however only provides for a rather course-grained whole-site replication for load balancing purposes. More appropriate is an application level mechanism [BMS96] for specifying which documents of what servers are replicated, because several groups have found that generally only a small subset of pages of a server is very popular. One more advantage of this approach is that replication can be made visible to users knowing best which documents are important to them. With an appropriate interface, they can specify the documents to be kept up to date with the original data, e.g. over night in order to save the annoyingly often occurring and long-lasting cache validation checks during document access (IMS). Such a system may be another step towards application-level quality of service features for Internet services.

Further Issues

Various further topics exist that appear to be worthwhile to be considered in this context. Especially in the area of inter-cache communication techniques, different concepts have been proposed, like using multicasting to disseminate informations [X, KnG95] as well as locate replicated data [X, MLB95]. Due to the inherent non-scalability and the lack of a widely available infrastructure for multicasting on the Internet, these approaches are currently of no practical use. More important however, are two other topics raised, resp. sharpened by employing caches for document forwarding. One is caused be the question of who may cache which data because of privacy considerations. The other deals with the legitimate desire for document providers to find out about the popularity of their work.

Privacy, Security and Authentication

HTTP 1.1 offers new options to explicitly specify which documents may be cached at which sites. Cache-Control directive private for example only permits client-local caches to store the data, whereas public permits any cache to retain the document. This presumes shared caches to be trustworthy entities, an assumption at least questionable in today's Internet. This also applies to the no-store cache control directive forcing intermediate caches not to keep any information about an entity delivered through it. The only effective guard against malign interference by either caches or any other program in the path from server to client is encryption where secrecy is important, resp. secure hash sums where only the authenticity of the information is required. Such a protocol will never find its way into the lower level Internet protocols because they are not necessary often enough to justify the high overhead in processing time and communication bandwidth they induce. After all, if certain precious documents can only be cached in an encrypted form, the issue arises, whether it is worthwhile at all to cache such documents that are most probably only accessed (and paid for) by very few users. Alternatively, such documents can be propagated and stored in a dedicated, and commercially maintained replication infrastructure in which the storage such documents require can be accounted and paid for appropriately.

A completely different, although related problem hitherto only known to users of firewalls is the problem of host-based authentication. As long as requests are sent directly from client to server, it is principally, although not completely reliably, possible to control access to one's server by per-host or per-domain access control lists, a technique commonly found in WWW servers. As soon as a user accesses the data via a proxy, this proxy appears to be the origin of the request to the server, thus effectively circumventing such access control mechanisms. This problem together with the need for reliable authentication because of commercial transactions caused the introduction of application level authorization protocols. As already substantiated above, such protocols cannot find their way into the basic IP protocol suite.

Accounting

Since Web servers gained commercial significance, the hit rate, i.e., the overall access rate to a some given server, became the most important asset of a site. This is in direct contrast to the very purpose of caching, namely to serve requests before they reach the originating server. Because of this, many commercial document providers willingly defeat caching by specifying Expires headers with extremely low values, like the special value Now, or by generating all documents on the fly with dynamically created names that are not reused and therefore not cacheable. In order to solve this conflict of

interests, a mechanism should be integrated that allows caches to send -appropriately anonymized- statistical data to those origin servers interested in the requests the proxy has served on their behalf. The next generation caches could and should do this carefully with respect to bandwidth usage. Current implementations as for example in the Harvest system are rather crude in this respect. They can be configured to send an echo request to the originating server as soon as a document originating from that host is served directly by the cache. Thus, the feedback is provided outside the critical path of a request. However, originating servers must be configured to understand such requests as normal hits and count them accordingly. The bigger problem with this approach is that the network may be unduly stressed, and that the information may be lost at all, since using UDP packets is potentially lossy, rendering it unacceptable as a solution for document providers.

A more sensible approach would be the installation of a trusted network of caches that can be individually configured to send counts of (cache) hits of previously configured hosts offline, for example once per day in one compressed feedback transmission.

Future developments

As already briefly mentioned in the section on the technical foundations of caching in the World-Wide Web, the new standard version of the hypertext transfer protocol, 1.1, holds several new opportunities in store for enhanced caching and replication strategies. The possibility to create and maintain persistent, long-lived connections between proxy caches and between servers and proxy caches as put forth in the preliminary specification of a next-generation hypertext transfer protocol [X, W3C96] additionally appears to be promising for building more efficient caching infrastructures.

As far as the caching hierarchy itself is concerned, a new layer enabled by high speed, low latency LANs might find its way into Web document caching. As proposed for example in [X, DW+94] several caching proxies may use each other’s main memory to cooperatively create a very fast caching federation often devoid of slow disk accesses. The main obstacle to this vision becoming true is the still unclear notion of locality, i.e., the small set of most beneficially held WWW pages, an area clearly in need of further research.

Summary

In conclusion, one can state that many different approaches exist to solve the latency and bandwidth problems currently plaguing the Internet. These approaches, however, often have conflicting effects on one another. No one solution can therefore be applicable in any situation and no single mechanism can be built into the low-level Internet infrastructure. Although caching and replication strategies clearly seem to help reducing the symptoms of the problem, it remains questionable whether such techniques can remedy the overall issue. On the one hand, Web-inherent topics like sites willingly defeating caching strategies may be resolved by providing a hit-reporting infrastructure into tomorrow’s caches. On the other hand, however, applications adapting to available end-to-end bandwidths, e.g. video presentations or telephony over the Internet [X, CT+95], will again greedily fill all the bandwidth saved by whatever technical means and leave many parts of the Internet as congested as they are today. Therefore it is most important to closely study the end-to-end effects of applications like the World-Wide Web that use the Internet as its communications infrastructure. Some of these observations might lead to enhancements in the set of Internet protocols and services useful to

many applications. The provision of a quality of service guaranteeing replicated service location service might be one example.

References

In order to be able to make our continuously maintained and expanded list of references on work in the area of caching and replication in the World-Wide Web easily accessible, we are listing only the central sources of information used in this survey. The reference [X,n] as used in the paper refers to the n-th entry in an external document accessible via our WWW server at the URL http://www.uni-kl.de/AG-Nehmer/GeneSys/WLIS/Caching.html.

[BC+95] A. Bestavros, R.L. Carter, M.E. Crovella, C.R. Cunha, A. Heddaya, S.A. Mirdad: *Application-level Document Caching in the Internet*; Proc. 2nd Workshop on Services in Distributed and Networked Environments, SDNE'95, Whistler, Canada, June 1995.

[BMS96] M. Baentsch, G. Molter, P. Sturm: *Introducing Application-level Replication and Naming into today's Web*; Computer Networks and ISDN Systems, Vol. 28, No. 7; Proc. 5th International Conference on the World-Wide Web, Paris, May 1996.

[CD+96] A. Chankhunthod, P.B. Danzig, C. Neerdaels, M.F. Schwartz, K.J. Worrell: *A Hierarchical Internet Object Cache*; Proc. 1996 USENIX Winter Technical Conference, San Diego, Jan. 1996.

[GwS96] J.S. Gwertzman, M. Seltzer: *World-Wide Web Cache Consistency*; Proc. 5th USENIX 1996 Annual Technical Conference, San Diego, Jan. 1996.

[LuA94] A. Luotonen, K. Altis: *WWW Proxies*; Computer Networks and ISDN Systems, Vol. 27, No. 2; Proc. 1st International Conference on the World-Wide Web, May 1994.

[Smi96] N.G. Smith: *The UK national Web cache - The state of the art*; Computer Networks and ISDN Systems, Vol. 28, No. 7; Proc. 5th International Conference on the World-Wide Web, Paris, May 1996.

[X, n] Reference to n-th entry of online reference list maintained at http://www.uni-kl.de/AG-Nehmer/GeneSys/WLIS/Caching.html(#n).

X-Files

[AS+94] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams, E.A. Fox: *Caching Proxies: Limitations and Potentials*; Proc. 4th International Conference on the World-Wide Web, Boston, Dec. 1994.

[Be95a] A. Bestavros: *Using Speculation to Reduce Server Load and Service Time on the WWW*; Technical Report TR-95-003, Boston University, CS Department, Feb. 1995.

[Be95b] A. Bestavros: *Demand-based Resource Allocation to Reduce Traffic and Balance Load in Distributed Information Systems*; Proc. 7th IEEE Symposium on Parallel and Distributed Processing, San Antonio, Oct. 1995.

[BlA92] M. Blaze, R. Alonso: *Dynamic Hierarchical Caching in Large-Scale Distributed File Systems*; Proc. International Conference on Distributed Computing Systems, 1992.

[BoH96] J.-C. Bolot, P. Hoschka: *Performance engineering of the World-Wide Web: Application to dimensioning and cache design*; Computer Networks and ISDN Systems, Vol. 28, No. 6; Proc. 5th International Conference on the World-Wide Web, Paris, May 1996.

[Cla94] K.C. Claffy: *Web traffic characterization: an assessment of the impact of caching documents from NCSA's Web server*; Proc. 2nd International Conference on the World-Wide Web, Chikago, 1994.

[CT+95] Z. Chen, S. Tan, R.H. Campbell, Y. Li: *Real-time Video and Audio in the World-Wide Web*; Proc. 4th International Conference on the World-Wide Web, Boston, Dec. 1995.

[DiP96] A. Dingle, T. Partl: *Web cache coherence*; Computer Networks and ISDN Systems, Vol. 28, No. 7; Proc. 5th International Conference on the World-Wide Web, Paris, May 1996.

[DW+94] M.D. Dahlin, R.Y. Wang, T.E. Anderson, D.A. Patterson: *Cooperative Caching: Using Remote Client Memory to Improve File System Performance*; Proc. 1st Symposium on Operating Systems Design and Implementation (OSDI), 1994.

[FG+96] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee: *Hypertext Transfer Protocol -- HTTP/1.1*; Internet Draft, Aug. 1996, available at: http://www.w3.org/pub/WWW/Protocols/HTTP/1.1/draft-ietf-http-v11-spec-07.txt.

[Gla94] S. Glassman: *A caching Relay for the World Wide Web*; Proc. 1st International Conference on the World-Wide Web, Geneva, May 1994.

[GwS95] J.S. Gwertzman, M. Seltzer: *The Case for Geographical Push-Caching*; Proc. 5th Workshop on Hot Topics in Operating Systems, Orcas Island, Wa, May 1995.

[Har95] The Harvest Information Discovery and Access System; Online documents at http://harvest.cs.colorado.edu and at http://harvest.cs.colorado.edu/harvest/Cached-1.4-announce.txt.

[HK+88] J.H. Howard, M.L. Kazar, S.G. Menees, D.A. Nichols, M. Satyanarayanan, R.N. Sidebotham, M.J. West: *Scale and Performance in a Distributed File System*; ACM Transactions on Computer Systems, Vol.6, No.1, Feb. 1988.

[KnG95] J. Knight, S. Guest: *Using Multicast Communications to Distribute Code and Data in Wide Area Networks*; Software-Practice and Experience, Vol.25, No.5, May 1995.

[MLB95] R. Malpani, J. Lorch, D. Berger: *Making World Wide Web Caching Servers Cooperate*; Proc. 4th International World-Wide Web Conference, Boston, Dec. 1995.

[Moc87] P. Mockapetris: *Domain Names- Concepts and Facilities*; Internet Request for Comments (RFC) 1034; Online available at http://ds.internic.net/rfc/rfc1034.txt.

[MuH92] D. Muntz, P. Honeyman: *Multi-level caching in distributed filesystems -or - your cache ain't nuthin' but trash*; Proc. Winter USENIX Conference, January 1992.

[NLA96] National Laboratory for Applied Network Research: *A Distributed Testbed for National Information Provisioning*; Online documentation and measurements at http://www.nlanr.net/Cache, 1996.

[W3C96] World-Wide Web Consortium: *Hypertext Transfer Protocol - Next Generation*; Online reference at http://www.w3.org/pub/WWW/Protocols/HTTP-NG/Overview.html.

[WaC96] Z. Wang, J. Crowcroft: *Prefetching in World Wide Web*; Proc. IEEE Global Internet 96, London, Nov. 1996.

[Wes95] D. Wessels: *Intelligent Caching for World-Wide Web Objects*; Proc. INET'95, 1995.

[Wes96] D. Wessels: *Squid Internet Object Cache*; Homepage of the Squid project, http://www.nlanr.net/Squid/, 1996.