# Reduce, Reuse, Recycle:
# An Approach to Building Large Internet Caches


| Syam Gadde | Michael Rabinovich | Jeff Chase |
|---|---|---|
| Dept. of Computer Science | AT&T Labs - Research | Dept. of Computer Science |
| Duke University | Murray Hill, NJ 07974 | Duke University |
| Durham NC, 27708 | misha@research.att.com | Durham NC, 27708 |
| gadde@cs.duke.edu | | chase@cs.duke.edu |



## Abstract

*New demands brought by the continuing growth of the Internet will be met in part by more effective use of caching in the Web and other services. We have developed CRISP, a distributed Internet object cache targeted to the needs of the organizations that aggregate the end users of Internet services, particularly the commercial Internet Service Providers (ISPs) where much of the new growth occurs.*

*A CRISP cache consists of a group of cooperating caching servers sharing a central directory of cached objects. This simple and obvious strategy is easily overlooked due to the well-known drawbacks of a centralized structure. However, we show that these drawbacks are easily overcome for well-configured CRISP caches.*

*We outline the rationale behind the CRISP design, and report on early studies of CRISP caches in actual use and under synthetic load. While our experience with CRISP to date is at the scale of hundreds or thousands of clients, CRISP caches could be deployed to maximize capacity at any level of a regional or global cache hierarchy.*


## 1 Introduction

The Internet community is struggling to keep pace with continuing traffic growth on the Internet backbone and at popular Web sites.

One response is to attempt to anticipate demand increases and ratchet up service capacity to meet them. Today, server accelerator caches and clustered Web servers are common, but these solutions do not help to slow the growth of backbone traffic. On the other hand, site replication (e.g., mirroring) is still mostly manual and not easily hidden from users, although some progress has been made toward automating some aspects of it (most recently with geographical push-caching [7], document dissemination [5] and smart clients [12]). While these "supply-side" approaches contribute to scalability, we believe that the traditional "demand-side" approach — demand-driven caching initiated by clients at their own expense and for their own benefit — is still the most powerful and economically correct tool at our disposal.

The key to a successful demand-side approach is to encourage comprehensive use of caching within the organizations that provide Internet service to user communities. We refer to these endpoint organizations as ISPs, although they include academic and other institutions as well as commercial Internet Service Providers. ISPs have the strongest economic incentive to deploy caches: they want to provide faster and better service to their users with less bandwidth. In doing so, the ISP incidentally acts as a good citizen by reducing the load it places on Web servers and the Internet backbone.

Some ISPs already service their Web clients through shared proxy servers [9]. We believe that the most effective caches will be those serving the largest user communities. Our studies of typical Web usage at AT&T indicate that large shared proxy caches can dramatically increase the global hit rate seen by a community of 3806 users, even for objects referenced over a 24-hour period (see Section 3.1).

CRISP (**C**aching and **R**eplication for **I**nternet **S**ervice **P**erformance) is a new Internet caching service designed to serve the needs of ISPs with thousands or tens of thousands of users or more. The basic problem solved by CRISP is that large central proxy caches suffer from a variety of performance, scalability, and organizational problems. CRISP allows ISPs to construct very large distributed caches as collections of proxy servers, without sacrificing the benefits of a shared cache. A CRISP cache can be larger and support more users than a central proxy cache, and it can be expanded incrementally by adding new servers. CRISP proxies can be deployed at different points in the organization's network, close to the clients they serve.

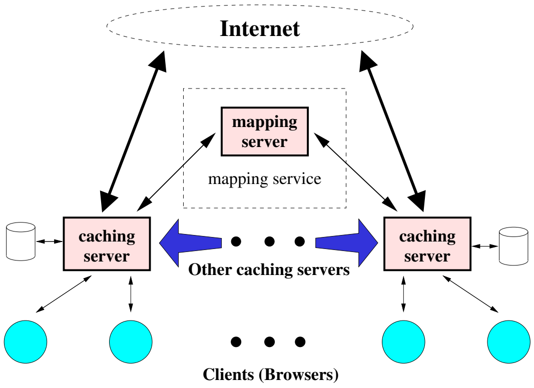


**Figure 1. A CRISP distributed proxy cache.**

CRISP servers cooperate to share their caches, using a central mapping service with a complete directory of the cache contents of all participating proxies. This obvious strategy is easily overlooked due to concerns that the central mapping service might become a bottleneck or a single point of failure. We argue below that these well-known concerns are not a problem for well-configured CRISP caches, given the properties of Web access. Moreover, the CRISP structure can simplify approaches to other issues raised by widespread deployment of large Internet caches, including consistency management, automated load balancing, alternative replacement strategies [11], and network-aware cache structure. For Internet caches, the simplest strategy for cooperative caching is the most effective as well as the easiest to implement and extend.

Section 2 presents an overview of the CRISP cache and the case for a central mapping service for distributed Internet caches. Section 3 presents some trace studies and early experiments, and Section 4 outlines our conclusions.

## 2 Architecture and Rationale

The CRISP cache has been implemented and is in active use at AT&T Labs. Our current prototype is built on top of the Harvest cache [6]. Figure 1 depicts the structure of a CRISP cache. Each client (e.g., a browser) is bound to one of several *caching servers* or *proxies*, which cache objects on behalf of their clients. Any URL fetch request that misses in a client's private browser cache (if any) is sent to the local proxy. The CRISP cache is cooperative: if the requested object is cached at any proxy in the cache, the object is delivered from that cache without accessing its home Internet site.

To probe the cooperative cache, the proxy forwards the requested URL to a *mapping server*. The mapping service maintains a complete cache directory; proxies notify the mapping service any time they add or remove an object from the cache. In this way, any proxy can probe the entire cache with a single unicast message exchange. If the map indicates that a requested object is resident in a peer, the requesting proxy retrieves the object directly from that peer, and returns it to the client. To the client, a CRISP cache appears exactly as a traditional proxy.

CRISP's use of a central mapping service distinguishes it from other cooperative Internet caches, e.g., the Harvest cache [6], its successor Squid [1], and Malpani et. al. [10]. In these systems, each proxy probes the cache by multicasting queries to all of its peers, to avoid relying on any single peer or a mapping server. If the request generates multiple hits, the requester can select its preferred site based on time to respond. Unfortunately, multicasting increases network traffic and forces all caches to respond to each request in most cases. In effect, every caching server incurs the load of cache probes that are limited to the mapping server in CRISP.

The Harvest cache and its predecessors [2] also support a *hierarchy* of caches, e.g., for regional or global caches. CRISP complements rather than replaces hierarchical caching. CRISP caches could be used to flatten the lowest levels of a hierarchical cache, or to incrementally expand capacity at any level of a hierarchy.

### 2.1 The Costs of a Central Map

The CRISP mapping service allows independent proxy caches (e.g., at the department or branch level) to be coalesced to form a single large cache. However, this central structure imposes new costs to store the central map, and message exchanges to probe the map and notify it of fetches or evictions at participating proxies.

At first approximation, the dollar cost of the mapping service is the price of a machine with enough physical memory to hold what amounts to a replica of the directories of all participating proxy caches. The map is simply a hash of URLs, which are small relative to the Internet objects they name. A recent study [3] indicated that the average object is about 10K bytes. If we conservatively estimate the average URL size at 100 bytes, then the data that passes through the mapping service is about 1% of the aggregate size of the CRISP cache, e.g., a gigabyte of storage for a 100G cache. However, we require that the map reside in physical memory since disk accesses would add unacceptably to the mapping server latency.

The central mapping service is not a latency bottleneck or a significant barrier to scalability for three basic reasons:

- Requests to the map contain little more than a URL. Since no actual object data passes through the mapping server, it does not have the I/O or network demands of a cache, and is therefore able to serve the needs of many caches without saturating.

- The response time of a well-configured and properly functioning mapping server is well below the ability of a human being to perceive the delay, and is small relative to the cost of retrieving the object itself, even for a cache hit (see Section 3).
- A CRISP mapping service can expand incrementally by adding more mapping servers, dividing responsibilities by statically partitioning the URL space across the servers. While a statically partitioned mapping service may suffer from hot spots, this can be offset by configuring mapping servers well below their saturation point, which adds only marginally to the cost of a CRISP cache.

The key limitation of CRISP is that the round-trip network latency between each proxy and the mapping service must be low, say, 20 ms at most. This may in practice limit the *geographical* size of the cache. In general, we envision that mapping service traffic will be carried as high-priority traffic over an ISP's internal network.

### 2.2 Handling Failures

Another potential drawback of the CRISP structure is that the mapping server is a central resource whose failure would render all or part of the cooperative cache temporarily unreachable. Also, the map could become out-of-date if a proxy server disconnects.

Because the objects cached by CRISP are read-only, failure of a mapping server *does not impair the ability of a CRISP proxy to service local cache misses* by fetching objects directly from their home sites until the map is operational again. Though the effectiveness of the cache is temporarily reduced, map failure does not result in denial of service. Similarly, the failure of a proxy is discovered lazily by its peers or the mapping server, and the maps are adjusted, perhaps after a few false hits or misses. For any kind of failure, the maps are restored using a simple recovery protocol, during which the cache operates correctly but with reduced efficiency.

If a mapping server believes that a proxy has failed, it marks all cache directory entries from that cache as *unavailable*. Clients that were connected to the failed proxy must fail over to a secondary proxy, e.g., by using autoconfiguration scripts [4]. When a proxy rejoins the CRISP cache, it first registers itself with the mapping service to re-enable or restore the maps for cached objects that survived the failure. Similarly, if a mapping server fails and recovers, proxies rebuild the central map by "rejoining" the CRISP cache, registering their current cache contents with the recovering mapping server.

Although failures are rare, the costs of recovery are acceptable. Assuming as above that the map is 1% of the size of the cache, the map for a gigabyte proxy can be transmitted in a few seconds over a T3 link. More importantly, a recovering mapping server can begin serving requests immediately, while it is still rebuilding its directories, although it may may return false misses until the maps are fully reconstructed. Effectiveness during recovery could be improved by a variety of techniques, e.g., registering the hottest objects first.

### 2.3 The Case for a Central Map

CRISP's simple and obvious design yields a cooperative Web cache that is easy to manage, reconfigure and expand incrementally. A CRISP cache can be expanded simply by registering more proxies with the mapping service. Clients can be redistributed across CRISP proxy servers to balance load without losing their ability to access any portion of the cache. This scalability and ease of configuration increases the likelihood that caches will be widely deployed and used effectively.

The central map can simplify approaches to important issues raised by large-scale caching in the Internet. A CRISP mapping service layer has complete knowledge of the cache contents, and the power to redirect requests to any participating proxy. If given the power to order transfers or evictions of cached objects, the mapping service could coordinate global replacement strategies, e.g., global LRU, which would simplify load balancing. As another example, scalable cache consistency could be implemented by using the mapping service as a single point of entry for invalidation directives issued from a modified object's home Web site.

## 3 Experiments

In this section, we present data to demonstrate the benefits of shared caches and study the properties of the CRISP architecture, using four traces of Internet access from different user communities. Our two largest traces are taken from the access logs from a centralized proxy cache used by thousands of the AT&T technical community, and a 24-hour subset (Aug. 29, 1996) of the publicly available proxy traces from Digital Equipment Corporation [8]. These traces are labeled *AT&T* and *DEC* in the figures, and represent accesses from 3806 and 3636 clients respectively. The *Medium* trace was taken from a proxy used by 218 researchers from one AT&T Bell Labs center. The fourth workload, *Small*, was taken from a pilot CRISP cache used by a group of 23 volunteers. The trace periods range from 24 hours for the largest traces to several days for the *Small* trace.

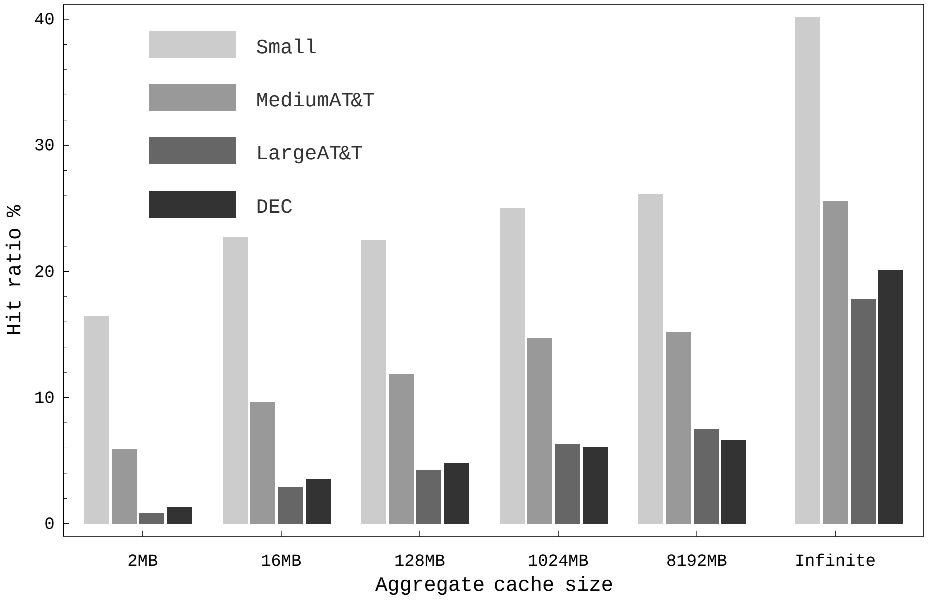


**Figure 3. Hit ratios for per-client caches**

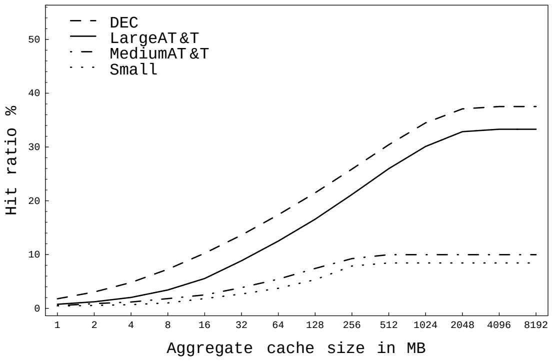


**Figure 2. Hits due exclusively to sharing**

## 3.1 Benefits of a Shared Proxy Cache

We first present results from trace-driven simulations to show the degree of sharing and temporal locality inherent in the recorded Web access traces. Our intent is to demonstrate the benefits of building large shared caches. For these experiments the caches used LRU replacement; alternative replacement policies tuned for Web access patterns would likely yield higher hit ratios [11]. CRISP caches could support alternative global replacement policies at the level of the mapping service.

The behavior of a shared cache is affected by the degree of temporal locality in the reference streams from individual clients. Temporal locality varies in the different traces, presumably because the references were filtered through private browser caches or lower-level proxies to varying degrees in the different user communities. Figure 3 shows the hit ratios yielded for each of the traces by a cache that is statically partitioned among the clients, i.e., by a cache of aggregate size $X$ shared by $N$ clients, each client having a private cache of size $X/N$. The rightmost group of bars shows the theoretical maximum hit ratio of per-client caches for each of the four traces; these are the hit ratios that would result from a partitioned cache of infinite size. This shows the degree of temporal locality inherent in the traces.

To illustrate the benefits of a shared cache, we first normalized the traces by filtering them through infinite per-client caches, then played the filtered traces into shared caches of various sizes. Figure 2 shows the resulting hit ratios for each of the four traces. These results are optimistic in the sense that the shared cache is isolated from the effects of temporal locality, i.e., the replacement policy is not biased by objects that were kept hot because of repeated references from a single client. However, the shared cache hit ratios reported in Figure 2 are conservative in three respects. First, all hits reported are due *exclusively to sharing of objects among different clients*. Second, a large, unified shared cache would likely yield additional temporal hits in practice, since per-client caches are not infinite, and the most active clients can receive a larger portion of the shared cache at any given time. Third, to the extent that the traces are filtered through lower-level proxies, the hit ratios in Figure 2 do not reflect hits resulting from sharing among clients using the same lower-level proxy.

We make the following observations:

- *Temporal locality varies in the traces.* The workloads from the *Small* trace experienced the highest overall hit ratios due to repeated references to non-shared objects from individual clients. This is because most of the volunteers restricted or disabled their private browser caches in order to test the experimental CRISP cache. The relatively low temporal hit ratio in the large traces reflects the filtering of the reference streams by browser caches and lower-level proxies.

- *Small caches are useless for sharing.* The largest traces exhibit the lowest hit ratios for small cache sizes, because objects live in the cache for only a short time before they are evicted. Only large caches yield the full benefit of shared object references from multiple clients.
- *More users, more sharing.* Given sufficiently large caches, traces from the larger user communities exhibited the highest peak hit ratios, and the greatest benefit from a shared cache. The benefit to smaller user communities is more modest.

The DEC trace yielded a 37% hit ratio due exclusively to sharing, illustrating the benefit of a shared cache in this instance. In fact, hits on shared objects accounted for around 60% of all hits in this trace, giving strong support for cooperative caching of Internet objects.

The shared hit ratios reported in Figure 2 do not show the effect of duplicate objects in CRISP caches. In practice, a CRISP cache yields slightly lower hit ratios than a central shared cache of the same size. This is because objects shared among clients bound to different proxy servers are duplicated in both proxies in our current prototype, reducing the effective size of the CRISP cache. This is the cost of the CRISP architecture's ability to support larger shared caches serving larger user communities than would be possible with a single central proxy. This cost could be reduced by more aggressive replacement of duplicates, under the control of mapping server, or perhaps by careful assignment of clients to proxy servers in the distributed CRISP cache.

### 3.2 Scalability

We placed CRISP caches under worst-case load from synthetic clients to evaluate the scalability of CRISP's central mapping server. For these experiments, we ran the CRISP cache on a cluster of 266MHz AlphaStation 500 workstations linked by a switched 100 Mb/s ethernet. The clients continuously fetch distinct small objects, so no accesses are intercepted by proxy hits. This results in a probe to the mapping server for every request. The synthetic nature of this workload is irrelevant because the mapping server is not affected by any access properties other than the request arrival rate and URL length.

Table 3.2 shows the average reponse time for a mapping server query in a CRISP configuration with six proxies and varying request arrival rates. The first two columns represent the worst-case and average number of clients that would generate the query rate shown in the third column. The first column (*conservative*) assumes that all clients surf the Web as fast as they can position and click the mouse (five seconds). The second column (*average*) gives the number of clients needed to generate the given query rate, assuming each client emits requests at the average per-client request rate observed in the largest AT&T trace at the time of peak load.

Table 3.2 shows that the mapping server response time, including local network delay, is well below human perception even under the expected peak load from 85,000 average clients.

| *clients (conserv.)* | *clients (average)* | *query rate (reqs./sec)* | *average latency* |
|---|---|---|---|
| 1605 | 84935 | 321 | 2.15ms |
| 1480 | 78320 | 296 | 1.79ms |
| 895 | 47363 | 179 | 1.46ms |
| 470 | 24872 | 94 | 1.45ms |

**Table 1. Mapping server query latency.**

## 4 Summary and Conclusions

Internet caches serve a dual purpose. First, caches minimize access latencies for objects that are shared by multiple clients or accessed repeatedly by a single client. Second, caches improve performance for all users by reducing load on the network backbone and servers. While private browser caches can filter repeated accesses, only shared caches can eliminate the need for a site to deliver a given object to every client that requests it, helping to slow traffic growth and absorb bursts in demand for popular sites.

CRISP allows construction of very large distributed proxy caches to serve the needs of Internet-connected organizations (ISPs, commercial or otherwise). This is important because demand-side caches are a critical part of the solution for scalability problems in the Internet. ISPs have the incentive and resources to deploy caches to improve service to their members or customers. In this way, the cost of expanded capacity is borne by those who benefit from it the most.

We have argued that CRISP's simple implementation of cooperative caching, based on a central mapping service, is effective and efficient for distributed Web caches serving tens of thousands of users. In particular, the properties of Internet object access (read-only access to relatively large objects with small URLs, tolerance of latencies below human perception, and the ability to fall back to the Internet if the map fails) allow us to safely enjoy the simplicity and benefits of the centralized structure.

## Acknowledgments

We wish to thank Fred Douglis, Tony DeSimone, and Sandeep Sibal for many stimulating discussions of caching issues and for valuable feedback. We would also like to thank Brian Larkins, David Kristol, and Doug Monroe for providing us with blinded access traces from AT&T proxies, and Jeff Mogul, Tom Kroeger, Carlos Maltzahn, and Digital Equipment Corporation for providing the same from DEC proxies. Thanks also go to the CRISP user volunteers and Darrell Anderson, Carla Ellis, Carmine Greco, H. V. Jagadish, Alvin Lebeck, Joe Shamblin, and Ken Yocum for offering comments on early drafts.

## References

[1] Squid Internet Object Cache. URL:http://squid.nlanr.net/.

[2] R. Alonso and M. A. Blaze. Dynamic hierarchical caching for large-scale distributed file systems. In *Proceedings of the USENIX Summer Conference*, June 1992.

[3] M. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 1996.

[4] Netscape Communications Corporation. Netscape Navigator 3.0 — Automatic Proxy Configurations. Available at <URL: http://home.netscape.com/comprod/products/navigator /version_3.0/management/proxy/index.html>, 1997.

[5] A. Bestavros, R. Carter, M. Crovella, C. Cunha, A. Heddaya, and S. Mirdad. Application level document caching in the Internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.

[6] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A hierarchical Internet object cache. In *USENIX 1996 Annual Technical Conference*, January 1996.

[7] J. Gwertzman and M. Seltzer. The case for geographical push-caching. In *Fifth Workshop on Hot Topics in Operating Systems*, 1995.

[8] T. M. Kroeger, J. Mogul, and C. Maltzahn. Digital's web proxy traces. Available at<URL: ftp://ftp.digital.com /pub/DEC/traces/proxy/webtraces.html>.

[9] A. Luotonen and K. Altis. World-wide Web proxies. First International Conference on the World-wide Web, 1994. Available at <URL:http://www.cern.ch/Papers /WWW94/luotonen.ps>.

[10] R. Malpani, J. Lorch, and D. Berger. Making World Wide Web caching servers cooperate. In *Fourth International World-wide Web Conference*, pages 107–117, December 1995.

[11] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox. Removal policies in network caches for World-wide Web documents. In *ACM SIGCOMM '96: Applications, Technologies, Architectures and Protocols for Computer Communication*, August 1996.

[12] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler. Using smart clients to build scalable services, June 1996. To appear in *USENIX 1997 Annual Technical Conference*.