Web Proxy Caching

“Proxy servers” interpose a cache between the client and the origin site for each object.

• deliver objects at LAN speeds
• mask network and server failures
• reduce traffic on servers and the network backbone
• reduce cost of network bandwidth for ISPs

Hit ratios peak at about 50% for Web caches.

...enough to stay a few months ahead of traffic growth...
...but miss penalty is a critical factor in performance.
The good news: 10% of the trace size yields 90% of potential hits.
Making Web Caches Scalable

Proxy hit ratios tend to grow logarithmically with client population.

- more users share more documents
- pattern varies with population

How can we build Internet caches powerful enough to serve very large user populations?

- Build distributed Web caches that share cached content among a group of proxies.

  How should peer servers coordinate?
Distributed Web Caches

Distributed, cooperative, collective
Issues for Web Caching

• binding clients to proxies, handling failover
  manual configuration, router-based “transparent caching”,
  Microsoft WPAD (Web Proxy Automatic Discovery)

• object-appropriate cache management
  dynamic documents, HTTP 1.1 cache control, caching
  responses vs. caching documents, lazy vs. eager or “push”

• effect of network topology
  How does the structure of the network affect the structure of
  the distributed cache?

• coordination among peer proxy servers
  Use a cache sharing/routing protocol (ICP, CARP) or
  distributed cache directory (CRISP)?
Web Cache Consistency

“Requirements of performance, availability, and disconnected operation require us to relax the goal of semantic transparency.”
- HTTP 1.1 specification

- expiration
  Origin server may add a “freshness date” response header, or the cache may determine expiration time heuristically.

- validation
  Cache may issue a conditional GET (including a validator header) if the cache entry may have expired.
  
  server specifies weak or strong validator (e.g., modify time)

- warning
  Cache may/must add “may be stale” warnings to headers.
Hierarchical Caches

**Idea**: place caches at exchange or switching points in the network, and cache at each level of the hierarchy.

- origin Web site (e.g., U.S. Congress)
- upstream
- downstream
- clients
- clients
- clients

INTERNET
Content-Sharing Among Peers

*Idea:* Since siblings are “close” in the network, allow them to share their cache contents directly.
Harvest-Style ICP Hierarchies

INTERNET

Examples
Harvest [Schwartz96]
Squid
NAC NetCache

Idea: multicast probes within each “family”: pick first hit response or wait for all miss responses.

For a cache level with \( n \) siblings, inter-sibling ICP traffic (and aggregate overhead) is quadratic with \( n \); sibling query handling overhead grows linearly with \( n \).
Cache Array Routing Protocol (CARP)

Examples
Microsoft Proxy Server
Netscape “proxy autoconfig”

INTERNET

“GET www.hotsite.com”

hash function

Advantages
1. single-hop request resolution
2. no redundant caching of objects
3. allows client-side implementation
4. no new cache-cache protocols
Issues for CARP

• no way to exploit network locality at each level
  – e.g., relies on local browser caches to absorb repeats

• load balancing
  • hash can be balanced and/or weighted with a load factor reflecting the capacity/power of each server
  • must rebalance on server failures
    – Reassigns \(\frac{1}{n}\)th of cached URLs for array size \(n\).
    – URLs from failed server are evenly distributed among the remaining \(n-1\) servers.

• miss penalty and cost to compute the hash
Summary Cache for ICP

_Idea_: each caching server replicates the cache directory ("summary") of each of its peers (e.g., siblings).

- Query a peer only if its local summary indicates a hit.
- Servers update their summaries lazily.
  
  Update when "new" entries exceed some threshold percentage.
  Update delays may yield false hits and/or false misses.

- To reduce storage overhead for summaries, implement the summaries compactly using _Bloom Filters_.
  
  May yield false hits (e.g., 1%), but not false misses.
  Each summary is three orders of magnitude smaller than the cache itself, and can be updated by multicasting just the flipped bits.
A Summary-ICP Hierarchy

Summary caches at each level of the hierarchy reduce inter-sibling miss queries by 95+%. What are the limits to scalability? 
...if we grow the cache sizes?
...if we grow the number of peers?
...if we deepen the hierarchy?

How significant are the gains relative to pure ICP or CARP...or nothing?

e.g., Squid configured to use cache digests

INTERNET

client

hit

miss
The Really Big Picture

(borrowed from http://ircache.nlanr.net)
Web Caching in the Large

Local/regional networks might be trees, but they connect through NAPs to backbones, which are intertwined at network exchange points.

On the backbone, parent/child relationships depend on the location of the client and origin site.

“Siblings” may have weak network connectivity.

deep “hierarchies” increase:
- miss costs
- storage overhead
Cache Routing in Squid

Squid allows non-tree “hierarchies”.

- Use origin site domain name to direct miss forwarding among “mutual parents”.
  Ideally, always forward toward the origin server.
- Use domain filtering to constrain sibling queries.
- Cache configuration is manual and tricky.

Squid is used for the NLANR national caching infrastructure over the vBNS backbone.
vBNS is an NSF-funded experimental backbone $3B funding starting in 1994

interconnects:
- US supercomputer centers
- US public NAPs (MAEs)
- Cary (NC-GNI)
- Duke through OC-48 ring

-provided by MCI 1/95
-coordinated by NLANR
-2.4 Gbps access (OC-48)
-SONET backbone
Many caches are “mutual parents”: miss forwarding is filtered by domain.
Embed Caching in the Network?

• **Adaptive Web Caching** [Floyd, Zhang, Jacobson]
  Self-configuring cache meshes: use IP routing information to forward cache misses toward origin servers.
  Should we combine ICP with HTTP?

• **Future**: provide caching as NAP service?
  Can we make it transparent/translucent?

• Could have NAPs or backbone switching centers access nearby caches [Legedza/Guttag]
  Store cache directories at the switching centers?
  Whose cache do you trust?
  Who pays for external access to a third-party cache?
The CRISP Architecture

**Key idea:** maintain a global cache directory independent of the caches themselves.

The key issue for caches is the structure of the distributed directory service; many organizations are possible.
Partitioned Synchronous Directory (PSD)

INTERNET

“GET www.cs.duke.edu”

client

mapping server(s)

a-g

h-o

p-z

(hash function simplified)

Degenerate case: one partition is Central Synchronous Directory.
Replicated Asynchronous Directory (RAD)

“GET www.cs.duke.edu”

INTERNET

e.g., Summary Cache

asynchronous directory updates
Replicated Partial Directory (RPD)

“GET www.cs.duke.edu”

INTERNET

RPD is a compatible with summary cache, e.g., replicate only entries with “proven sharing value”, or those that are “nearby” (vicinity cache).

client

asynchronous directory updates

Only partial directories propagated to caching servers
Member servers have overlapping *vicinities*; cache hits are delivered only from within the vicinity.

Popular objects diffuse through the mesh.