Large-Scale Web Caching and Content Delivery

Jeff Chase

CPS 212: Distributed Information Systems
Fall 2000
Caching for a Better Web

Performance is a major concern in the Web

Proxy caching is the most widely used method to improve Web performance

- Duplicate requests to the same document served from cache
- Hits reduce latency, network utilization, server load
- Misses increase latency (extra hops)

[Source: Geoff Voelker]
Cache Effectiveness

Previous work has shown that hit rate increases with population size [Duska et al. 97, Breslau et al. 98]

However, single proxy caches have practical limits
  • Load, network topology, organizational constraints

One technique to scale the client population is to have proxy caches cooperate

[Source: Geoff Voelker]
Cooperative Web Proxy Caching

Sharing and/or coordination of cache state among multiple Web proxy cache nodes

Effectiveness of proxy cooperation depends on:

- Inter-proxy communication distance
- Size of client population served
- Proxy utilization and load balance

[Source: Geoff Voelker]
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)

INTERNET

Resolve misses through the parent.

upstream

downstream

clients

DUKE Systems & Architecture
Content-Sharing Among Peers

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

Examples
Harvest [Schwartz96]
Squid (NLANR)
NetApp NetCache

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

• With ICP: query traffic within “families” (size $n$)
  Inter-sibling ICP traffic (and aggregate overhead) is quadratic with $n$.
  Query-handling overhead grows linearly with $n$.

• miss latency
  Object passes through every cache from origin to client: deeper hierarchies scale better, but impose higher latencies.

• storage
  A recently-fetched object is replicated at every level of the tree.

• effectiveness
  Interior cache benefits are limited by capacity if objects are not likely to live there long (e.g., LRU).
Hashing: Cache Array Routing Protocol (CARP)

INTERNET

Microsoft Proxy Server

Advantages
1. single-hop request resolution
2. no redundant caching of objects
3. allows client-side implementation
4. no new cache-cache protocols
5. reconfigurable
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 \((1/n)\)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
  In CARP, hash cost is linear in \(n\): hash with each node and pick the “winner”.
Directory-based: Summary Cache for ICP

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

[Cao et. al. Sigcomm98]

- Query a peer only if its local summary indicates a hit.
- 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+%.

e.g., Squid configured to use cache digests

INTERNET

client

summary caches

miss
hit

object request
object response
query
query response
Issues for Directory-Based Caches

- Servers update their summaries lazily.
  
  Update when “new” entries exceed some threshold percentage.

  Update delays may yield false hits and/or false misses.

- Other ways to reduce directory size?
  
  Vicinity cache [Gadde/Chase/Rabinovich98]

  Subsetting by popularity [Gadde/Chase/Rabinovich97]

- What are the limits to scalability?
  
  If we grow the number of peers?

  If we grow the cache sizes?
On the Scale and Performance....

[Wolman/Voelker/.../Levy99] is a key paper in this area over the last few years.

- first negative result in SOSP (?)
- illustrates tools for evaluating wide-area systems simulation and analytical modeling
- illustrates fundamental limits of caching benefits dictated by reference patterns and object rate of change
  forget about capacity, and assume ideal cooperation
- ties together previous work in the field wide-area cooperative caching strategies
  analytical models for Web workloads
- best traces
# UW Trace Characteristics

<table>
<thead>
<tr>
<th>Trace</th>
<th>UW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>7 days</td>
</tr>
<tr>
<td>HTTP objects</td>
<td>18.4 million</td>
</tr>
<tr>
<td>HTTP requests</td>
<td>82.8 million</td>
</tr>
<tr>
<td>Avg. requests/sec</td>
<td>137</td>
</tr>
<tr>
<td>Total Bytes</td>
<td>677 GB</td>
</tr>
<tr>
<td>Servers</td>
<td>244,211</td>
</tr>
<tr>
<td>Clients</td>
<td>22,984</td>
</tr>
</tbody>
</table>

[Source: Geoff Voelker]
A Multi-Organization Trace

University of Washington (UW) is a large and diverse client population

   Approximately 50K people

UW client population contains 200 independent campus organizations

   Museums of Art and Natural History
   Schools of Medicine, Dentistry, Nursing
   Departments of Computer Science, History, and Music

A trace of UW is effectively a simultaneous trace of 200 diverse client organizations

   • Key: Tagged clients according to their organization in trace

[Source: Geoff Voelker]
Cooperation Across Organizations

Treat each UW organization as an independent “company”
Evaluate cooperative caching among these organizations

How much Web document reuse is there among these organizations?
- Place a proxy cache in front of each organization.
- What is the benefit of cooperative caching among these 200 proxies?

[Source: Geoff Voelker]
Ideal Hit Rates for UW proxies

Ideal hit rate - infinite storage, ignore cacheability, expirations

Average ideal local hit rate: 43%

[Source: Geoff Voelker]
Ideal Hit Rates for UW proxies

Ideal hit rate - infinite storage, ignore cacheability, expirations

Average ideal local hit rate: 43%

Explore benefits of perfect cooperation rather than a particular algorithm

Average ideal hit rate increases from 43% to 69% with cooperative caching

[Source: Geoff Voelker]
Sharing Due to Affiliation

UW organizational sharing vs. random organizations
Difference in weighted averages across all orgs is ~5%

[Source: Geoff Voelker]
Cacheable hit rate - same as ideal, but doesn’t ignore cacheability

Cacheable hit rates are much lower than ideal (average is 20%)

Average cacheable hit rate increases from 20% to 41% with (perfect) cooperative caching

[Source: Geoff Voelker]
Scaling Cooperative Caching

Organizations of this size can benefit significantly from cooperative caching. 

But…we don’t need cooperative caching to handle the entire UW population size:
  - A single proxy (or small cluster) can handle this entire population!
  - No technical reason to use cooperative caching for this environment
  - In the real world, decisions of proxy placement are often political or geographical

How effective is cooperative caching at scales where a single cache cannot be used?

[Source: Geoff Voelker]
Hit Rate vs. Client Population

Curves similar to other studies
  - [e.g., Duska97, Breslau98]

Small organizations
  - Significant increase in hit rate as client population increases
  - The reason why cooperative caching is effective for UW

Large organizations
  - Marginal increase in hit rate as client population increases

[Source: Geoff Voelker]
In the Paper...

1. Do we believe this? What are some possible sources of error in this tracing/simulation study?
   What impact might they have?

2. Why are “ideal” hit rates so much higher for the MS trace, but the cacheable hit rates are the same?
   What is the correlation between sharing and cacheability?

3. Why report byte hit rates as well as object hit rates?
   Is the difference significant? What does this tell us about reference patterns?

4. How can it be that byte hit rate increases with population, while bandwidth consumed is linear?
Trace-Driven Simulation: Sources of Error

1. *End effects*: is the trace interval long enough?
   Need adequate time for steady-state behavior to become apparent.

2. *Sample size*: is the population large enough?
   Is it representative?

3. *Completeness*: does the sample accurately capture the client reference streams?
   What about browser caches and lower-level proxies? How would they affect the results?

4. *Client subsets*: how to select clients to represent a subpopulation?

5. Is the simulation accurate/realistic?
   cacheability, capacity/replacement, expiration, latency
What about Latency?

From the client’s perspective, latency matters far more than hit rate.

How does latency change with population?

Median latencies improve only a few 100 ms with ideal caching compared to no caching.

[Source: Geoff Voelker]
Questions/Issues

1. How did they obtain these reported latencies?
2. Why report median latency instead of mean?
   Is the difference significant? What does this tell us? Is it consistent with the reported byte hit ratios?
3. Why does the magnitude of the possible error decrease with population?
4. What about the future?
   What changes in Web behavior might lead to different conclusions in the future?
   Will latency be as important? Bandwidth?
Large Organization Cooperation

What is the benefit of cooperative caching among large organizations?

Explore three ways

- Linear extrapolation of UW trace
- Simultaneous trace of two large organizations (UW and MS)
- Analytic model for populations beyond trace limits

[Source: Geoff Voelker]
Extrapolation to Larger Client Populations

Use least squares fit to create a linear extrapolation of hit rates.

Hit rate increases logarithmically with client population, e.g., to increase hit rate by 10%:
  - Need 8 UWs (ideal)
  - Need 11 UWs (cacheable)

“Low ceiling”, though:
  - 61% at 2.1M clients (UW cacheable)

A city-wide cooperative cache would get all the benefit.

[Source: Geoff Voelker]
UW & Microsoft Cooperation

Use traces of two large organizations to evaluate caching systems at medium-scale client populations

We collected a Microsoft proxy trace during same time period as the UW trace

- Combined population is ~80K clients
- Increases the UW population by a factor of 3.6
- Increases the MS population by a factor of 1.4

Cooperation among UW & MS proxies…

- Gives marginal benefit: 2-4%
- Benefit matches “hit rate vs. population” curve

[Source: Geoff Voelker]
## UW & Microsoft Traces

<table>
<thead>
<tr>
<th>Trace</th>
<th>UW</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>7 days</td>
<td>6.25 days</td>
</tr>
<tr>
<td>HTTP objects</td>
<td>18.4 million</td>
<td>15.3 million</td>
</tr>
<tr>
<td>HTTP requests</td>
<td>82.8 million</td>
<td>107.7 million</td>
</tr>
<tr>
<td>Avg. requests/sec</td>
<td>137</td>
<td>199</td>
</tr>
<tr>
<td>Total Bytes</td>
<td>677 GB</td>
<td>N/A</td>
</tr>
<tr>
<td>Servers</td>
<td>244,211</td>
<td>360,586</td>
</tr>
<tr>
<td>Clients</td>
<td>22,984</td>
<td>60,233</td>
</tr>
<tr>
<td>Population</td>
<td>~50,000</td>
<td>~40,000</td>
</tr>
</tbody>
</table>

[Source: Geoff Voelker]
UW & MS Cooperative Caching

Is this worth it?

[Source: Geoff Voelker]
Use an analytic model to evaluate caching systems at very large client populations

- Parameterize with trace data, extrapolate beyond trace limits

Steady-state model

- Assumes caches are in steady state, do not start cold
- Accounts for document rate of change
- Explore growth of Web, variation in document popularity, rate of change

Results agree with trace extrapolations

- 95% of maximum benefit achieved at the scale of a medium-large city (500,000)

[Source: Geoff Voelker]
Inside the Model

[Wolman/Voelker/Levy et. al., SOSP 1999]
  • refines [Breslau/Cao et. al., 1999], and others

Approximates asymptotic cache behavior assuming Zipf-like object popularity
  • caches have sufficient capacity

Parameters:
  • $\lambda$ = per-client request rate
  • $\mu$ = rate of object change
  • $p_c$ = percentage of objects that are cacheable
  • $\alpha$ = Zipf parameter (object popularity)
Zipf

[Breslau/Cao99] and others observed that Web accesses can be modeled using *Zipf-like probability distributions*.

- Rank objects by popularity: lower rank $i \rightarrow$ more popular.
- The probability that any given reference is to the $i$th most popular object is $p_i$
  
  Not to be confused with $p_c$, the percentage of cacheable objects.

Zipf says: “$p_i$ is proportional to $1/i^\alpha$, for some $\alpha$ with $0 < \alpha < 1$”.

- Higher $\alpha$ gives more skew: popular objects are *way* popular.
- Lower $\alpha$ gives a more *heavy-tailed* distribution.
- In the Web, $\alpha$ ranges from 0.6 to 0.8 [Breslau/Cao99].
- With $\alpha=0.8$, 0.3% of the objects get 40% of requests.
Cacheable Hit Ratio: the Formula

$C_N$ is the hit ratio for cacheable objects achievable by a population of size $N$ with a universe of $n$ objects.

$$C_N = \int_1^n \frac{1}{C^x} \left( \frac{1}{\mu C^x} + \frac{1}{\lambda N} \right) dx$$

$$C = \int_1^n \frac{1}{x^\alpha} dx$$
Inside the Hit Ratio Formula

Approximates a sum over a universe of \( n \) objects...

...of the probability of access to each object \( x \)...

...times the probability \( x \) was accessed since its last change.

\[
C_N = \int_1^n \frac{1}{C x^\alpha} dx
\]

\[
1 \leq C_N = \int_1^n \frac{1}{\mu C x^\alpha} \frac{1}{1 + \frac{\mu C x^\alpha}{\lambda N}} dx
\]

\[
C = \int_1^n \frac{1}{x^\alpha} dx
\]

\[
C = 1/\Omega
\]

in [Breslau/Cao 99]

\[
0 < \alpha < 1
\]

\( C \) is just a normalizing constant for the Zipf-like popularity distribution, which must sum to 1. \( C \) is not to be confused with \( C_N \).
Inside the Hit Ratio Formula, Part 2

What is the probability that \( i \) was accessed since its last invalidate?

\[
= \frac{\text{rate of accesses to } i}{\text{rate of accesses or changes to } i}
\]

\[
= \frac{\lambda N p_i}{\lambda N p_i + \mu}
\]

\[
C_N = \int_1^n \frac{1}{C x^\alpha} \left( \frac{1}{1 + \frac{\mu C x^\alpha}{\lambda N}} \right) dx
\]

Divide through by \( \lambda N p_i \).

Note: by Zipf \( p_i = 1/Ci^\alpha \)

so: \( 1/(\lambda N p_i) = Ci^\alpha/\lambda N \)
Hit Rates From Model

Cacheable Hit Rate

- Focus on cacheable objects

Four curves correspond to different rate of change distributions

- Believe even Slow and Mid-Slow are generous

Knee at 500K – 1M

[Source: Geoff Voelker]
Extrapolating UW & MS Hit Rates

These are from the simulation results, ignoring rate of change (compare to graphs from analytic model).

What is the significance of slope?
Latency From Model

Straightforward calculation from the hit rate results

[Source: Geoff Voelker]
Rate of Change

What is more important, the rate of change of popular objects or the rate of change of unpopular objects?

Separate popular from unpopular objects
Look at sensitivity of hit rate to variations in rate of change

[Source: Geoff Voelker]
Rate of Change Sensitivity

Popular docs sensitivity
- Top curve
- Unpopular low R-of-C
- Issue is minutes to hours

Unpopular docs sensitivity
- Bottom curve
- Popular low R-of-C
- Days to weeks to month

Unpopular more sensitive than popular!
- Compare differences in hit rates between A,C and B,C

[Source: Geoff Voelker]
Hierarchical Caches and CDNS

What are the implications of this study for hierarchical caches and Content Delivery Networks (e.g., Akamai)?

- Demand-side proxy caches are widely deployed and are likely to become ubiquitous.
- What is the marginal benefit from a supply-side CDN cache given ubiquitous demand-side proxy caching?
- What effect would we expect to see in a trace gathered at an interior cache?

CDN interior caches can be modeled as upstream caches in a hierarchy, given some simplifying assumptions.
An Idealized Hierarchy

Assume the trees are symmetric to simplify the math.
Ignore individual caches and solve for each level.
**Hit Ratio at Interior Level $i$**

$C_N$ gives us the hit ratio for a complete subtree covering population $N$.

The hit ratio predicted at level $i$ or at any cache in level $i$ over $R$ requests is given by:

\[
\frac{\text{hits at level } i}{\text{requests to level } i} = \frac{h_i}{r_i} = \frac{R p_c (C_{N_i} - C_{N_{i+1}})}{r_{i+1} - h_{i+1}}
\]

“the hits for $N_i$ (at level $i$) minus the hits captured by level $i+1$, over the miss stream from level $i+1$”
Root Hit Ratio

Predicted hit ratio for cacheable objects, observed at root of a two-level cache hierarchy (i.e. where $r_2 = R_{p_c}$):

$$\frac{h_1}{r_1} = \frac{C_{N_1} - C_{N_2}}{1 - C_{N_2}}$$
Generalizing to CDNs

Symmetry assumption: $f$ is stable and “balanced”.
Hit ratio in CDN caches

Given the symmetry and balance assumptions, the cacheable hit ratio at the interior (CDN) nodes is:

\[
\frac{C_{N_I} - C_{N_L}}{1 - C_{N_L}}
\]

\(N_I\) is the covered population at each CDN cache. 
\(N_L\) is the population at each leaf cache.
Cacheable interior hit ratio

Interpretation: Interior hit rates improve as leaf populations increase.

Graphical representation:
- **X-axis:** Population served by each interior cache $[N_I]$
- **Y-axis:** Cacheable hit ratio
- **Legend:**
  - $N_I/N_L$ values: 65536, 16384, 4096, 1024, 256, 64, 16, 4

Conclusion:
Increasing $N_I$ and $N_L$ → Improved cacheable hit ratio
Interior hit ratio as percentage of all cacheable requests

marginal cacheable hit ratio

.... but, the interior cache sees a declining share of traffic.

increasing $N_I$ and $N_L$ -->