

CS290i - Lecture 17

Proxy Caches

Scalable Internet Services and Systems, Spring 2001

Thorsten von Eicken
Department of Computer Science
University of California at Santa Barbara

Proxies

† Types

- † Explicit
- † Transparent
- † Reverse

† Proxy Purposes

- † Firewall
- † Caching
- † Filtering
- † Access control
- † Load balancing

2

Caching issues

† Example: Squid 2.0

† When checking the object freshness, Squid calculates these values:

- † OBJ_DATE is the time when the object was given out by the origin server
- † OBJ_LASTMOD is the time when the object was last modified
- † OBJ_AGE is how much the object has aged since it was retrieved
- † LM_AGE is how old the object was when it was retrieved
- † LM_FACTOR is the ratio of OBJ_AGE to LM_AGE
- † CLIENT_MAX_AGE is the (optional) maximum object age the client will accept
- † EXPIRES is the (optional) expiry time from the server reply headers

† Refresh algorithm parameters:

- † URL regular expression
- † CONF_MIN: The time (in minutes) an object without an explicit expiry time should be considered fresh
- † CONF_PERCENT: A percentage of the objects age (time since last modification age) an object without explicit expiry time will be considered fresh
- † CONF_MAX: An upper limit on how long objects without an explicit expiry time will be considered fresh.

3

Squid example(cont.)

† Refresh Algorithm:

```

† if (EXPIRES) {
    if (EXPIRES <= NOW) return STALE
    else return FRESH
}
if (CLIENT_MAX_AGE)
    if (OBJ_AGE > CLIENT_MAX_AGE)
        return STALE
if (OBJ_AGE > CONF_MAX) return STALE
if (OBJ_DATE > OBJ_LASTMOD) {
    if (LM_FACTOR < CONF_PERCENT) return FRESH
    else return STALE
}
if (OBJ_AGE <= CONF_MIN)
    return FRESH
return STALE

```

4

Cache Hierarchy & ICP

† Idea

- † Cooperative caches to improve hit rates & decrease latency
- † Multiple caches to achieve scalability

† ICP

- † UDP-based protocol
 - † Use round-trip times & loss rates to load-balance among caches
- † “Do you have URL xyz cached?”
- † If yes, fetch document using HTTP
 - † Request may be denied

5

Do proxies scale?

† “On the scale and performance of cooperative web proxy caching”, A. Wolman, G. Volker, N. Sharma, N. Cardwell, A. Karlin, H. Levy, 1999.

- † Subtext: is all this research into scalable proxies worth it?

† Caching proxy analysis

- † Univ. Washington trace: “200 diverse organizations”
- † Microsoft trace
- † Analytical model

† Findings

- † Cooperative caching works for small orgs
- † Does not scale to larger orgs/populations
- † No point designing highly scalable coop caching schemes
- † Must increase document cacheability first

6

Traces & Cache Simulation

† Characteristics

- † UW
 - † 50000 students, faculty, & staff
 - † Snooping at border routers
- † MS
 - † 40000 employees
 - † At proxies
- † Collected simultaneously
 - † May 7 - 14, 1999
- † Anonymized, but classified by organization

Parameter	UW	Microsoft
HTTP Requests	82.8 million	107.7 million
HTTP Objects	38.4 million	15.3 million
Total Request Bytes	677 Gb	200M
Average Requests/Sec	173	199
Clients	22,064	30,233
Servers	244,211	360,586
Duration	7 days	6 days 6 hours

Table 1. Overall trace statistics.

† Simulation

- † Optimistic: Infinite cache size, no document expiration
- † Conservative: Cold start misses
- † 2 modes: practical (squid 2 rules) & ideal (everything deemed cacheable)

7

Population size

† Method

- † Choose random subset of clients
- † Error bars show different subsets

† Observations

- † More sharing at MS (by 13%)
- † Fraction of cacheable docs:
 - † 60% (UW), 51% (MS)
- † Knee at 2500 users
 - † Benefit in going from 200 to 2000 users
 - † Load could be handled by single cache

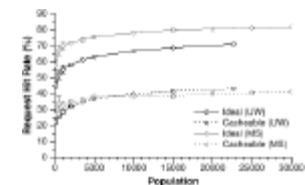


Figure 1. Proxy cache request hit rate as a function of client population.

8

Hit rate vs. latency

- † Observations
 - ‡ Little impact on latency
 - ‡ Many high-latency docs (mean>median)
 - ‡ Shared objects are smaller than others
 - ‡ Caching reduces bandwidth
 - † but no benefit to increased population

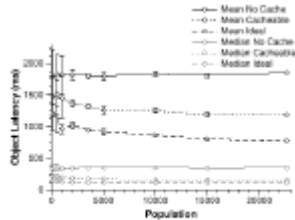


Figure 2. Mean and median request latency as a function of client population for the UW trace. The error bars on the median curves are the min and max medians across the slice.

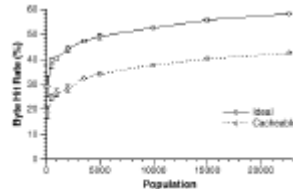


Figure 3. Proxy cache byte hit rate as a function of client population for the UW trace.

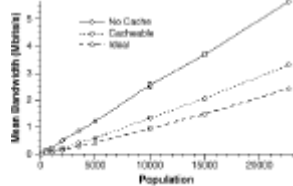


Figure 4. Bandwidth consumed as a function of client population for the UW trace.

Organizations

- † How does this translate to “real” organizations?
 - ‡ Use depts within UW as “orgs”
- † Methodology
 - ‡ Classify users into approx 200 orgs
 - ‡ Compare per-org cache with UW cache
 - ‡ Then cluster into 200 random orgs
 - ‡ Then perform clustering analysis to cluster users into 200 “optimal” orgs
- † Observations
 - ‡ Cooperation does improve hit rates
 - ‡ Random clustering is worse than UW orgs
 - † But only by 4%
 - ‡ Optimal clustering is little better than UW orgs
 - † By 1%

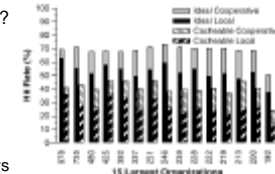


Figure 5. Breakdown of local and global proxy hit rates for the 15 largest UW organizations.

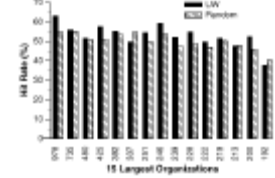


Figure 6. Comparison of the proxy hit rates for the 15 largest UW and randomly populated organizations.

Extrapolations

- † Curve fitting observations
 - ‡ UW: Little correlation between sharing and cacheability
 - ‡ MS: cacheable docs are less shared
 - ‡ Cacheable curves limited by cacheability
 - ‡ Ideal curves max out at 11M/2.9M users (UW/MS)
- † Sharing between UW and MS
 - ‡ Consider 100 most popular docs
 - ‡ 33% overlap
 - ‡ Consider cooperating proxies
 - ‡ 2%-5% hit rate increase
 - ‡ Unpopular docs universally unpopular
 - ‡ Coop helps popular docs only on first fetch

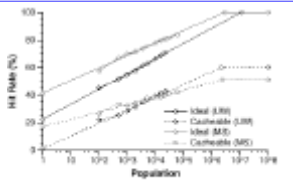


Figure 7. Proxy cache hit rate as a function of client population.

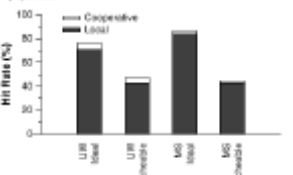


Figure 8. Hit rate benefit of cooperative caching between UW and Microsoft proxies.

Summary of trace analysis

- † 2500 users
 - ‡ Above 2500 users, little benefit to additional users
 - ‡ Four-folding UW population (using MS) increases hit rate by only 2.7%
 - ‡ Can be served by single proxy cache
 - ‡ Cooperating caches only useful to overcome administrative boundaries
 - ‡ No “special interest group” clustering
 - ‡ Groups formed though clustering analysis does not show better hit rates than UW organizations

Analytical model

† Model parameters

- † Number of clients
- † Total number of documents
- † Distribution of fraction of requests to document i (heavy tailed)
- † Average client request rate
- † Time between changes to a document
- † Probability that a doc is cacheable
- † Average document size
- † Last-byte latency of origin server

† Observations

- † Hit rate ~ request rate / change rate
- † Change rate of unpopular docs important
- † Large population can tolerate higher change rates

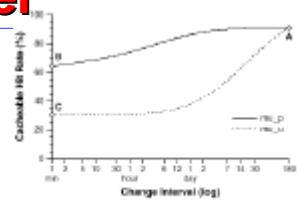


Figure 11. Sensitivity of hit rate to the rate of change of popular and unpopular documents, μ_p and μ_u . The hit rates were calculated for a population of 200,000.

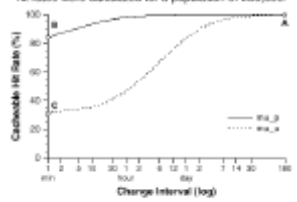


Figure 12. Sensitivity of hit rate to the rate of change of popular and unpopular documents, μ_p and μ_u . The hit rates were calculated for a population of 20 million.

Cooperation Schemes

† Three schemes

- † Hierarchical
- † Hash-based
 - † Clients hash URL to cache
- † Directory-based
 - † Caches broadcast directory deltas periodically

† Population sizes

- † "City": 500'000
- † "State": 5M
- † "West coast": 50M

† Observations

- † Level-1 caches have similar req rates
- † Hierarch: top-level becomes bottleneck
- † Hash-based stores doc only once

	Hierarchical	Hashed	Directory
Arrival Rate	$r_1 = 150M/day$ $r_2 = 50M/day$	50M/day	15M/day
Latency	0.85 secs	0.85 secs	0.89 secs
Storage	11 TB	1.5 TB	9.3 TB

Table 5. City cooperative caching performance.

	Hierarchical	Hashed	Directory
Arrival Rate	$r_1 = 1.5TB/day$ $r_2 = 14TB/day$ $r_3 = 29.5M/day$	29.5M/day	37.3M/day
Latency	0.76 secs	0.85 secs	0.35 secs
Storage	11 TB	150 GB	9.5 TB

Table 6. State cooperative caching performance.

	Hierarchical	Hashed	Directory
Arrival Rate	$r_1 = 1TB/day$ $r_2 = 1.5TB/day$ $r_3 = 14TB/day$ $r_4 = 29.5M/day$	29.5M/day	37.3M/day
Latency	0.76 secs	1.1 secs	1.11 secs
Storage	11 TB	15 GB	9.5 TB

Table 7. West Coast cooperative caching performance.

Conclusions

- † No need for highly-scalable cooperative caching schemes
 - † Any reasonable scheme works for sizes where cooperation is useful
- † Largest benefit of cooperative caching is to small organizations
 - † Grow user population beyond administrative boundaries
- † Performance overall is limited by cacheability of docs
 - † Focus research there
- † Little benefit to cooperating with others based on mutual interests of users
 - † Marginal benefit observed in UW/MS traces
- † Fundamental changes in web workloads can change these results
 - † E.g. streaming multimedia