Measuring the Performance of Prefetching Proxy Caches

Brian D. Davison
davison@cs.rutgers.edu

Department of Computer Science
Rutgers, The State University of New Jersey

The Problem

- **Traffic Growth**
  - User growth (~100M US users in 1999), More time online per user, Increasing connection speeds
- **Performance varies because of**
  - Heterogeneous network connectivity
    - From 28.8kbps w / 100ms latency (modem) to 1.5mbps w / 10ms latency (T1) to 192mbps w / 3ms latency (OC3)
  - Real world distances
    - From next door to thousands of miles across oceans or continents
  - Hot spots in WWW
    - Planned (Netscape 4.5), Unplanned (breaking news stories)
- **Biggest user complaint: need more speed**

One Solution: Web Caching

- Caching offers:
  - reduced bandwidth usage
  - reduced server load
  - faster responses
- A proxy cache accepts requests for web objects
  - Attempts to fill those requests from its cache
  - Fetches directly when not in cache
- Proxy caches are usually shared by many users and are often placed near network gateways to reduce network bandwidth.
- Prefetching can be used to further improve cache latencies.

Why Evaluate?

How can you find the best caching system for your network?

- How do you know if your new caching method is better than existing methods?
- How much improvement will clients notice with proxy caching?

Current Evaluation Methodologies

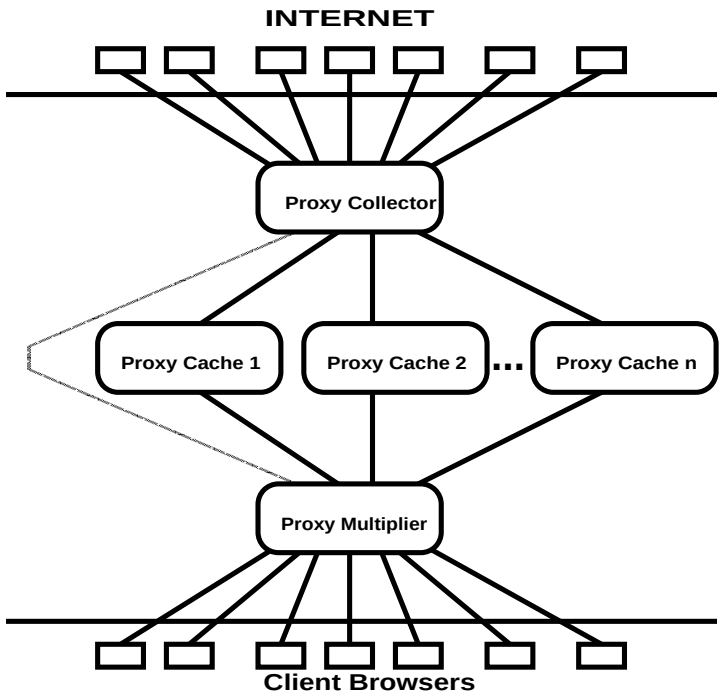
- **How to evaluate:**
  - Caching algorithms are simulated on a benchmark log of object requests.
  - Implemented systems are tested on artificial workloads with similar to real logs (e.g. same object sizes and recurrence rates).
  - Object / byte hit rates can be calculated and latencies estimated.
- **Difficulties:**
  - Simulation requires detailed knowledge (not always possible).
  - Comparing implemented systems as black-boxes helps, but often entails applying artificial request loads on an isolated network.
  - Captured trace logs reduce uncertainties with artificial logs, but since the ave. lifetime of a web page is short, captured logs lose value quickly as more objects referenced in it change or become inaccessible.
  - Cache logs are not always recorded accurately for playback.
  - Logs often captured for client populations different from target.

Prefetching to Improve Client Performance

- While the client may appreciate the indirect benefits of reduced bandwidth and server loads, reduced latencies are the most visible advantage of caching.
- To further improve client latencies, prefetching is often proposed in an attempt to retrieve objects in advance of a client request.
- Caches that prefetch on the basis of the contents of the pages being served, such as CacheFlow and Wcol need at least to have access to the links within web pages -- something that is not available from logs.
- Even if page contents were logged, caches that perform prefetching may prefetch objects that are not on the user request logs and thus have unknown characteristics of size and retrieval costs.

ROPE Architecture Overview

- To measure opaque systems, we use full implementations instead of simulation
- To allow for content-based prefetching and to capture current web characteristics, we suggest the use of a live network connection
- To avoid drawbacks of simulated or captured request logs, we suggest the use of a live request stream



ROPE Architecture Detail

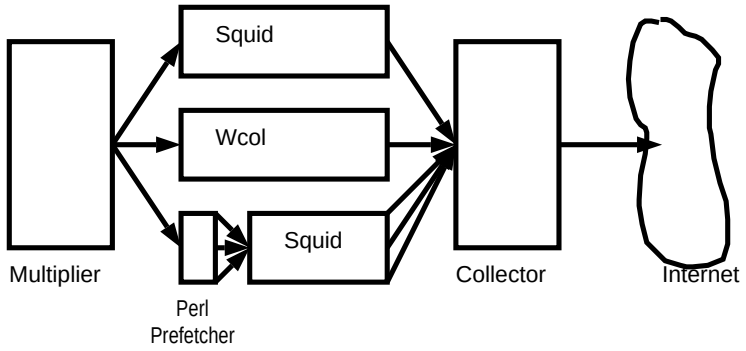
- Since live connections prevent replication of tests, we use simultaneous evaluation of multiple proxies so that comparisons can be made
- All client requests are sent to a single proxy (multiplier), which sends a copy to each proxy under evaluation.
- Proxy misses are passed on to another proxy (collector). The collector insures that only one request is sent to the origin server (by caching *all* responses)
- By forming a wrapper around each proxy, the ROPE architecture can accurately measure network usage and response times (if the collector is modified to return all responses as slowly as a miss).

Implementation

- **Multiplier only**
  - Approximately 1800 lines of code
  - In Java; eight classes
  - Multithreaded design, with timeouts
  - Supports HTTP 1.0 Requests
  - Random first proxy selection to send request
  - Sends uncachable requests directly
- **Uses standard proxy instead of collector**

Example ROPE Application: Bandwidth Savings

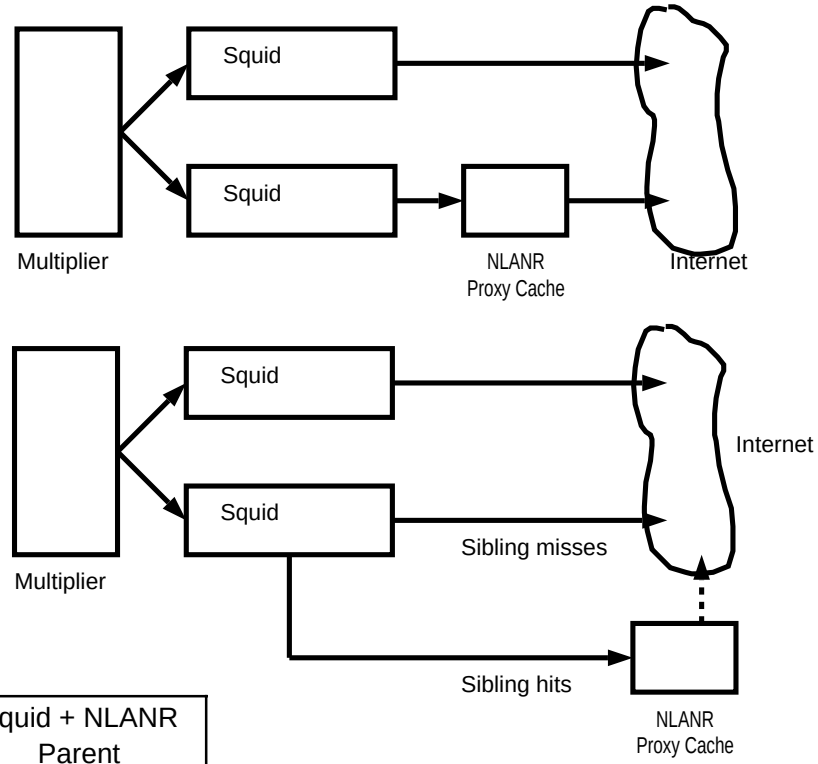
- **Evaluated three black-box proxies (100M cache each)**
  - Squid 1.1
  - Wcol (WWW Collector, prefetches first 10 links)
  - Squid + Perl Prefetcher (prefetches all links)
- **Playback of log over 24 hours**
- **10% of requests generated 400-class errors**
- **Collector was another Squid with 200M cache**



	Squid	Wcol	Squid+Prefetching
page hit rate	21.3%	37.7%	32.4%
byte hit rate	40.9%	52.5%	52.3%
bandwidth used	59%	>250%	>600%

Example ROPE Application: Latency Improvement

- **Does the use of a second-level proxy improve latency?**
- **No collector used**
- **Two smaller Squid caches (20M); one used NLANR cache**
- **Two versions of experiment: NLANR Cache as Parent or as Sibling proxy**



	Squid Alone	Squid + NLANR Sibling	Squid Alone	Squid + NLANR Parent
Ave. Squid latency	2.89s	2.96s	1.90s	2.16s
Ave. ROPE latency	2.81s	2.77s	1.80s	2.22s

Benefits of ROPE Architecture

- takes existing proxy-caches and uses them as black-boxes for evaluation purposes
- tests the proxy-caches on real-world data
- eliminates variation in measured performance due to network / server changes over time
- eliminates variation in performance due to dated request logs
- can evaluate prefetching systems that might fetch pages never seen by the user (and thus not included in a traditional trace)
- allows for the evaluation of content-based prefetching systems without having to locally mirror all pages referenced in a trace
- simultaneously compares proxies on the same request stream

Drawbacks of ROPE Architecture

- evaluating multiple prefetching schemes may place a higher demand for bandwidth on the network
- multiple copies of comparable hardware may be needed
- multiplier and collector may add additional latencies or bottlenecks
- there are many potential parameters to change (such as choosing an appropriate cache size)
- multiplier hides the client ip (as it may be useful for a proxy to perform tasks differently depending on the client)
- caches are evaluated independently (hierarchies are not included)
- tests are not replicable (applicable to any method using a live network connection)

Conclusions & Future Work

- **ROPE is complementary to existing methods**
  - The ROPE architecture has been designed in particular for use in an online environment with real traffic.
  - However it can still be useful as an evaluation technique with artificial workloads or trace logs.
  - Artificial workloads are useful for evaluating systems under workloads that cannot be captured or exceed current loads.
  - Can capture the statistical properties of specialized workloads and retain benefits of simultaneous evaluation on a live network.
- **Desired Future Work**
  - Live tests
  - Implementation of full collector proxy

For more information about this work

Send e-mail to davison@cs.rutgers.edu to request:

- Preprint of submission to SIGMETRICS'99, entitled "The Rutgers Online Proxy Evaluation (ROPE) Architecture"
- Preprint of submission to WWWC8, entitled "A Survey of Proxy Cache Evaluation Techniques"

Web caching resources:

http://www.cs.rutgers.edu/~davison/web-caching/