A Caching Relay for the World Wide Web

Steven Glassman

Systems Research Center
Digital Equipment Corportion
130 Lytton Ave.
Palo Alto, CA 94301
steveg@src.dec.com
http://www.research.digital.com/SRC/people/Glassman_Steve/bio.html

Abstract

We describe the design and performance of a caching relay for the World Wide Web. We examine how the behavior of users affects the performance of the relay. From an analysis of the data from the relay, we develop a model that describes some aspects of the Web as a whole. (A version of this paper is available at http://www.research.digital.com/-SRC/people/Glassman_Steve/-CachingTheWeb.html)

1 Overview

In January 1994, we set up a caching World Wide Web [10] relay for Digital Equipment Corporation's facilities in Palo Alto, California. We use a relay to reach the Web because Digital has a security firewall that restricts direct interaction between Digital internal computers and machines outside of Digital. We added caching to the relay because we wanted to improve the relay's performance and reduce its external network traffic. Clients use the relay for accessing the Web outside of Digital; requests for internal Digital pages go directly to the servers. When the relay has a current version of a Web page in its cache, it handles the request locally without ever contacting the remote server.

The relay handles about 2000 requests from 60 users each day. To date, it has served over 100,000 HTTP [3] and Gopher [2] requests from 300 users for 40,000 different pages. FTP [1] (file transfer) requests are handled by an existing (non-caching) FTP relay. We don't allow WAIS [9] (Wide Area Information System) or NNTP [6] (network news transfer protocol) requests through the relay.

A caching relay has several potential advantages over a non-caching relay; it:

- Reduces latency on requests for cached pages
- Reduces overall network load and remote server load
- Provides availability when a remote server is unavailable

And possible disadvantages are that it:

- Returns stale versions of a page when the remote version has changed and the cache doesn't know
- Increases latency on requests for non-cached pages
- Increases local administrative complexity and cost for disk space

We found that 30-50% of all requests are satisfied from the cache (a cache hit). When there is a cache hit, the relay supplies the page in 1/4 to 1/6 the time of retrieving the page over the network. The reduction in network load is equivalent to the cache hit rate. As described below, we did not find any of the disadvantages to be significant.

2 Our cache design

We based our cache design on studies of the logs from an existing Digital Web relay. We ran a simulation of our cache design using these logs to estimate the cache size, the cache hit ratio, and the effectiveness of various heuristics for estimating the validity of pages in the cache.

All cached Web pages are stored as Unix files, with the file name formed from the URL [8]. We plan for our cache to hold up to 80,000 Web pages. Since pages have an average size of 20K-25K bytes, we expect our cache size to reach about

2 GBytes. Rather than use a single giant directory with 80,000 files in it, we hash the file name into one of 4,096 sub-directories so that each directory holds about 20 files. The sub-directories are organized into a 3 level deep hierarchy where each internal directory has 16 sub-directories. For example, the URL http://www1.cern.ch/WWW94/-Abstracts/GNA.html is cached under the file name www1.cern.ch%%WWW94%%Abstracts%%-GNA.html. The hashed value of the URL is 5C7 (hex) and so the file is found in the directory /www/cache/5/C/7.

In our original design, the cache was entirely protocol-independent because this was simpler and we weren't yet sure how many different protocols we would have to handle. Each page was cached under a name based on a 128-bit fingerprint of the *entire* request. This meant that cache hits occurred on textually identical requests, but not on non-identical but equivalent requests. Therefore the cache was not getting hits when it should. This problem led us to abandon being protocol-independent, since the parsing of the requests to determine equivalency is protocol-dependent.

Our relay only handles two protocols - Gopher and HTTP. The Gopher protocol is very simple and a simple approach suffices. HTTP requests are more complex and must be parsed, but understanding the additional fields of the request and the response assists in caching.

The main problem in designing the cache is to avoid returning a stale page from the cache when a newer version of the page is available on the remote server. Neither HTTP nor Gopher has any provision for a server notifying the cache when a page changes, so the cache must estimate a time period during which it believes the page will not change.

A cached page is termed *valid* if the cache believes that it has the newest version (i.e. it believes the page hasn't changed on the remote server). A cached page *expires* when the page's valid time period ends (although expired pages remain in the cache). A *stale* page is a valid page that has in fact changed (i.e. the cache's estimate for the valid period was wrong).

If a request comes in for a valid page, the cache returns its version without contacting the remote server. If a request comes in for an expired page, the cache *refreshes* it from the remote server. Depending on the remote server, a page is refreshed either by retrieving an entire new copy or by checking whether the page has changed and retrieving a new version if it has.

The cache's algorithm for estimating the valid period of a page must balance two concerns. If the estimate is too long, then the probability of returning stale data increases. If the cache's estimate is too short, then the cache will have fewer valid pages and thus a lower cache hit rate.

The HTTP protocol allows for an "Expires" time in the response. Ideally, the cache could use this value to set the valid period for the page in the cache and never risk returning a stale page. Unfortunately, we have found only 6 pages (out of 28,000) that have an expiration time on them (and 5 are from the finger gateway). Therefore, we use the "Last modified" time, which is provided in nearly all HTTP responses, as a simple estimate of when the page will change next. For example, if a page has not changed in a week, we assume it will not change in the next week. If there is no modification time (the Gopher protocol doesn't give one), we use a default of 6 hours. We limit every page to be valid for at least one hour and no more than two weeks.

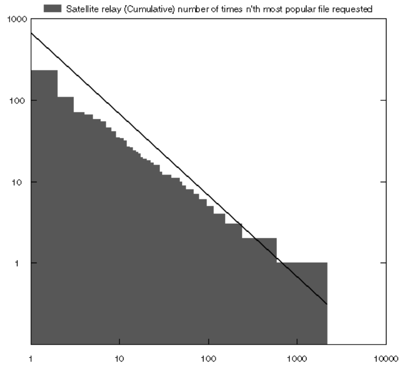
When an expired page is requested, we refresh it and calculate an expiration time as above. However, if there is no "Expires" or "Last modified" time, we compare the new version with the expired one that is still in the cache. If the page has not changed, we use the time of the expired version as the "Last modified" time and increase the valid period. If the page has changed, we reduce its valid period.

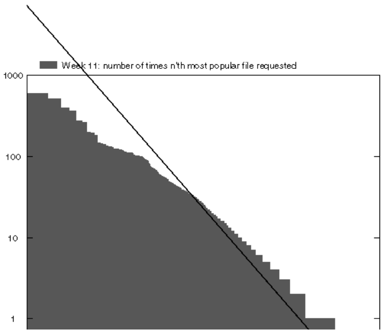
HTTP requests allow "Accepts" fields that list the acceptable MIME [4] (multimedia) formats of the response. (This is another reason why we must parse HTTP requests.) A cached page that is not in an acceptable MIME format cannot be returned. Fortunately, the HTTP specification is very generous about when a format is acceptable. The cache returns a cached page if it matches *any* acceptable format (and since most requests will accept any format, matching the "Accepts" is usually trivial). We have yet to find a cached page that was not in a format acceptable to a request.

One of our original goals for the cache was to minimize the external network load of the relay. We therefore did not implement any strategies such as pre-fetching (retrieving pages from the server speculatively) or refreshing cache pages as they expire. Now that we have data from operating the cache, we plan to examine these possibilities and compare the costs of the extra network load with the benefits of improved cache performance.

3 Satellite relays

Our caching relay can also be used as a *satellite* relay. A satellite relay is a secondary relay usually located at a site with poor connectivity or bandwidth to the net. It helps by providing a cache near the users, so they have access to cached pages at local network speeds.


Satellite relay (Cumulative) number of times n'th most popular file requested
1000
100
10
1
1
10
100
1000
10000


Week 11: number of times n'th most popular file requested
1000
100
10
1