Web Caching and Content Distribution: A View From the Interior*

Syam Gadde, Jeff Chase
Dept. of Computer Science
Duke University
Durham NC, 27708
{gadde,chase}@cs.duke.edu

Michael Rabinovich
AT&T Labs - Research
108 Park Avenue
Florham Park, NJ 07932
misha@research.att.com

Abstract

Research in Web caching has yielded analytical tools to model the behavior of large-scale Web caches. Recently, Wolman et al. have proposed an analytical model and used it to evaluate the potential of cooperative Web proxy caching for large populations.

This paper shows how to apply the Wolman model to study the behavior of *interior* cache servers in multi-level caching systems. Focusing on interior caches gives a different perspective on the model's implications, and it allows three new uses of the model. First, we apply the model to large-scale caching systems in which the interior nodes belong to third-party content distribution services. Second, we explore the effectiveness of content distribution services as conventional Web proxy caching becomes more prevalent. Finally, we correlate the model's predictions of interior cache behavior with empirical observations from the root caches of the NLANR cache hierarchy.

1 Introduction

As the Web continues to evolve, the most popular Web sites receive an increasing share of Internet traffic. These sites have a competitive motivation to employ advanced content distribution schemes to offer better service to their clients at lower cost. There is a growing trend toward outsourcing content distribution to commercial hosting services such as Exodus, Digital Island, GlobalCenter, and others. Major hosting providers and related companies maintain *content distribution networks* (CDNs) that cache or replicate content as needed to meet demand from clients across a wide area. Examples of CDNs include Mirror Image [10], Sandpiper Footprint [11], and Akamai's FreeFlow [1].

CDNs offer compelling benefits to Web sites and other content providers even as the price of wide-area bandwidth declines. The CDN services maintain the networks and data centers to serve the content; these shared resources offer economies of scale and allow the service to dynamically adjust content placement, request routing, and capacity provisioning to respond to demand and network conditions. The CDN protects the content provider from unpredicted demand surges or unecessary costs for excess capacity. Recent extensions to CDN services are directed at serving media streams and other resource-intensive content.

We refer to caches in CDNs as *supply-side* caches because they act as agents of the content providers rather than of the clients, as in conventional *demand-side* Web proxy caching. Supply-side content distribution supplements demand-side Web proxy caches, which are rapidly becoming ubiquitous. Until recently, proxy caches were an optional service for users who voluntarily configured their browsers to redirect requests through a proxy. Today, Internet Service Providers interpose caching transparently at the edges of the network to help meet the exponentially growing demand for Web-based services. The market for proxy caches and related software is now worth billions of dollars a year, and these caches serve tens

*This work is supported by the National Science Foundation under grant CCR-96-24857 and by the U.S. Department of Education GAANN program.

of millions of new clients who are not aware of their existence.

Caching CDNs supplement ubiquitous demand-side caches with a layer of upstream caches logically residing in the network interior. These interior caches behave similarly to upstream caches in demand-side *hierarchical* caching systems. A well-known example of hierarchical caching is the NLANR cache [14], an experimental large-scale Web cache that is a precursor to supply-side CDNs. CDNs differ from hierarchical caches primarily in that they generalize the *request routing* function used to direct the miss stream from lower-level (leaf) caches to the interior. Less significantly, CDNs may proactively populate interior caches, anticipating future requests.

Previous research [3, 6, 7, 8, 9] has studied demand-side benefits to cooperative caching in the Web by analyzing traces collected in specific caching environments. The goal of this paper is to adapt recent research results in demand-side Web caching to study the interaction of supply-side and demand-side approaches to content distribution. If demand-side caching is ubiquitous, what are the real benefits of supply-side content distribution from the standpoint of the service provider? What are the ranges of covered client populations for which supply-side content distribution is most effective?

Our study is based on an analytical model of steady-state caching behavior for static object requests in the Web. The model was proposed by Wolman et al. [15] as an extension to an earlier model developed by Breslau et al. [4] In their recent paper, Wolman et al. parameterized the model using large traces from a community of 83,000 clients at the University of Washington and Microsoft, then used the model to explore the limits of cooperation mechanisms for demand-side Web caches. This model is currently the best available tool to guide performance expectations for large-scale Web caching.

This paper makes the following contributions:

- It shows how to use the model to directly predict interior cache behavior given parameters such as client population sizes, document rates of change, and Zipf popularity parameters.
- It shows that supply-side caching CDNs can be modeled similarly to upstream caches in a hierarchical caching system, assuming a stable request routing function.
- It uses the extended model to explore the effectiveness of idealized CDNs under varying assumptions about the covered population and the populations served by the demand-side leaf caches.
- It correlates the model with empirical observations from the NLANR caching hierarchy. In this step, we use the model to predict population attributes of the NLANR hierarchy given traces and performance summaries from the root caching servers. This is significant because it shows how to use the NLANR experiment and its diverse user community to further validate the model and parameterize it more accurately.

The paper is organized as follows. Section 2 outlines the Wolman model and its significance, shows how to derive interior cache behavior from the model, and how to further apply our extended model to supply-side content distribution. Section 3 presents analytical results for interior cache hit ratios under a range of population assumptions. Section 4 presents empirical observations of root cache behavior in the NLANR hierarchy, and uses the model to derive the populations served by the NLANR cache. Section 5 concludes.

2 Modeling Interior Caches

We base our analysis on a model presented by Breslau et al [4], and extended by Wolman et al. [15] to incorporate document rate of change. Previous analytical and experimental work focuses on the behavior of complete caching systems from the client's perspective. This section derives new formulas to predict the behavior of interior caches deeper in the network, when the edges of the network already employ caching. This allows us to use the model to predict upstream cache behavior in common scenarios including caches at server sites, root servers in hierarchical caches, and interior caches in supply-side CDNs.

The Wolman model yields formulas to predict steady-state properties of static Web caching systems parameterized by population size, population request rate, document rate of change, size of the

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

$$C = \int_{1\le x\le n} \frac{1}{x^\alpha}dx$$

n = no. of objects N = population
α = Zipf parameter λ = request rate
μ = object rate of change

Table 1: The hit ratio formula from Wolman et al. C_N is the hit ratio for cacheable objects with a population of size N.

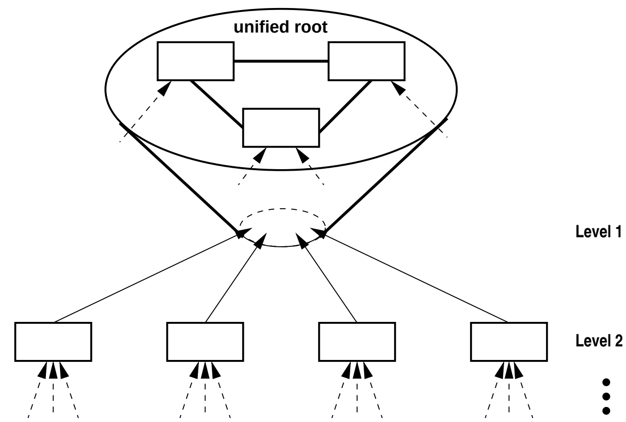


Figure 1: A typical cache hierarchy. Each level i receives r_i requests from level $i-1$. In the NLANR hierarchy, the root (detail) is actually several caches cooperating to act as a unified root.

object universe, and a popularity distribution for objects. The model makes several simplifying assumptions that make it tractable while preserving qualitative behavior: (1) object popularity follows a Zipf-like distribution as observed in [4], (2) object popularity is uniform across the population, (3) objects are cacheable with probability p_c independent of popularity, and (4) inter-arrival times for object requests and updates are exponentially distributed with a parameter determined by the object's popularity. The model does not account for capacity misses, thus its predictions give an upper bound on the effectiveness of the caching system as a whole. However, for our purposes the model is conservative in that it overstates the effectiveness of the leaf caches, and thus understates the potential incremental benefit of upstream interior caches or CDNs to supplement the leaf caches.

Table 1 summarizes the key formula from Wolman et al. yielding C_N, the aggregate object hit ratio for a population of size N considering only cacheable objects. The formula approximates a sum (over a universe of n objects) of the probability of access to each object multiplied by the probability that the object is resident in the cache, i.e., that it had already been accessed since its last change. The total hit ratio across all objects is given by $H_N = p_c C_N$.

We begin by applying the model to predict behavior of upstream caches in simple cache hierarchies, as illustrated in Figure 1. Each caching level i (the *root* is at level $i = 1$) is populated by one or more caches, each serving some client population. This structure is the basis for hierarchical proxy cache architectures, as introduced by the Harvest Internet Object Cache [5], and currently supported by its successors such as Squid [13] and some commercial caches.

In applying the model to this structure we idealize hierarchical caching systems in two key respects. First, we assume for now that the entire miss stream of a level $i+1$ cache is routed to the same cache in level i, i.e., each cache has at most one parent. Second, our application of the model in this paper assumes that each cache at any level i serves the same population size of N_i clients. This second assumption merely simplifies the analysis; the principles apply equally to more general hierarchies.

The Wolman et al. paper presents several formulas describing the aggregate behavior of idealized hierarchical caching systems similar to Figure 1. However, it does not directly predict hit ratios for the interior caches at each level. It is not difficult to derive these hit ratios from the model by applying it recursively at each level of the hierarchy, as described below.

Let R be the total number of requests presented by the client population over some sufficiently long time interval. Let r_i be the total number of requests seen at level i, with $r_i \le R$, and let h_i be the total number of hits seen at level i. Since the hit ratios for all caches at a given level are the same in the idealized hierarchy, it suffices to determine the aggregate hit ratio for all caches at that level acting together. The aggregate number of hits delivered by all caches at level i or below is given by $Rp_cC_{N_i}$; each level i cache covers a population N_i and therefore — together with its descendants — absorbs cacheable requests at the rate C_{N_i}. The number of hits delivered by each level i is this aggregate number of hits $Rp_cC_{N_i}$, minus the hits already absorbed by the level below it. Similarly, the total number of requests r_i seen at level i is equal to the number of misses forwarded from the level below, which is given by $r_{i+1} - h_{i+1}$. Therefore, the hit ratio at level i is given by:

$$\frac{h_i}{r_i} = \frac{Rp_c(C_{N_i} - C_{N_{i+1}})}{r_{i+1} - h_{i+1}}$$

For a two-level cache system, where $r_2 = R$, the hit ratio for the top-level cache is:

$$\begin{aligned} \frac{h_1}{r_1} &= \frac{Rp_c(C_{N_1} - C_{N_2})}{R - Rp_cC_{N_2}} \\ &= \frac{H_{N_1} - H_{N_2}}{1 - H_{N_2}} \end{aligned} \tag{1}$$

The hit ratio for *cacheable* requests at the root of the hierarchy (i.e. using $r_1 = Rp_c - Rp_cC_{N_2}$) reduces to the same equation but with C_x instead of H_x:

$$\frac{h_1}{r_1} = \frac{C_{N_1} - C_{N_2}}{1 - C_{N_2}} \tag{2}$$

We can also account for a percentage p_u of uncacheable requests that are not forwarded from lower-level caches because they are immediately recognized as uncacheable at request time:

$$\begin{aligned} \frac{h_i}{r_i} &= \frac{Rp_c(C_{N_i} - C_{N_{i+1}})}{R - h_{i+1} - (1-p_c)(1-p_u)r_{i+1}} \\ \frac{h_1}{r_1} &= \frac{H_{N_1} - H_{N_2}}{1 - H_{N_2} - (1-p_c)(1-p_u)} \end{aligned} \tag{3}$$

Note that Equation 3 reduces to Equation 1 when p_u is 100%. Most of our analysis focuses on cacheable requests, but Section 4 uses Equation 3 to correlate the model with real-world observations from the NLANR hierarchy.

2.1 Modeling Supply-Side CDNs

We now apply the model to derive interior cache behavior in more general multi-level caches. In particular, we model supply-side CDNs as a set of interior nodes in a two-level caching system, as illustrated in Figure 2. In this configuration, the lower-level or *leaf* nodes are demand-side caches managed by Internet Service Providers, acting as agents of the client population. The upstream nodes are controlled by the CDNs.

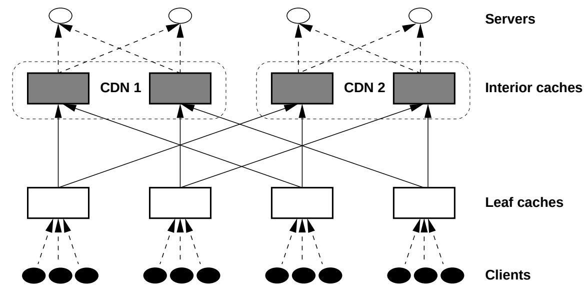


Figure 2: An example of a two-level caching system with CDNs in the interior.

This structure generalizes the simple hierarchy by introducing a *request routing function* that selects an interior cache to service each miss from a leaf cache. The routing function may consider the source of the request, the identity of the requested content, and a variety of factors including the load on the interior caches, proximity of the interior caches to the leaf, and network congestion. The routing function might be implemented in the leaf caches (as in the NLANR cache hierarchy described in Section 4), or in servers managed by the CDN, e.g., as described below. To identify the requested content, the routing function may switch on the entire request URL, the origin server IP address, or on some or all of the domain name embedded in the request URL.

Some CDNs, including Akamai Freeflow and Sandpiper Footprint, transparently interpose the routing function by placing it in the authoritative Domain Name Server for the domain containing the requested content. Since the CDN acts as an agent of the content provider, it may control the DNS service for the domains used by the content provider. To service a miss, the leaf cache first issues a DNS request for the IP address corresponding to the domain named in the request URL. The CDN's modified DNS server responds with the IP address of the selected interior cache, rather than the IP address of the origin server. The leaf cache then forwards the miss to the selected interior cache, believing that it is the origin site serving the requested content. A major advantage of this scheme is that it works without modifying the leaf caches.

One drawback of this DNS-based approach is that the routing function cannot distinguish among content objects in the same domain. This is because the leaf cache includes only the domain name in the DNS request, and not the entire URL. To obtain a finer-grained routing function, CDNs may provide replication tools that rewrite the URLs returned by the content provider, replacing the content provider's domain with any of a set of domains controlled by the CDNs. These tools also allow the content provider to specify which objects are cached or replicated by the CDN, and which are served directly by the origin server.

The effect of the request routing function is to concentrate requests for each object in a selected subset of the interior nodes. Thus each interior node may cover a larger population with respect to the objects that it serves, potentially capturing hits on shared objects that leaf caches serving smaller populations could not absorb. The CDN may use the routing function to suppress redundant copies of objects in order to conserve storage in the interior. In contrast, any interior node in a simple hierarchical cache might hold a copy of the object if one or more of its descendants requested the object in the recent past. With a routing function that maps every object to a unique interior cache, the system can deliver all possible hits if the *aggregate* size of the interior caches is sufficient to store the universe of objects, as in hash-based proxy caches [12]. A CDN will use additional storage to increase the degree of replication for specific domains or sets of objects, in order to reduce latency or distribute the load for serving those objects. These choices can be made dynamically by adjusting the request routing function to respond to observed load and network conditions.

The essential observation to make about this structure is that each interior cache serves some set of objects for some aggregate covered population whose requests are filtered through leaf caches with known populations. Thus we can apply the model directly to derive steady-state hit ratios for each set of objects at the interior caches, if we assume that the request routing function is stable, i.e., all requests for an object x from a particular leaf cache are consistently routed to the same interior cache. In practice, the results apply if the interval between request routing adjustments is sufficiently large for the steady-state behavior to become apparent.

Three additional simplifying assumptions allow us to derive a single formula that applies to any interior cache:

- For each object x assigned to a given interior cache c, c sees all of the requests for x from a population subset of a fixed size N_I — the clients of those leaf caches that direct misses for x to cache c.
- The subset of objects assigned to each interior cache shows a popularity distribution that is representative of the object universe. Specifically, we assume that the miss stream is divided among the CDNs and interior caches in proportion to the number of objects assigned to each.
- Every client's request stream passes through a leaf cache, and each leaf cache serves a population subset of a fixed size N_L.

With these assumptions, the hit ratio for cacheable objects in the interior caches (or in each CDN as a whole) is given directly by adapting Equation 2:

$$\frac{C_{N_I} - C_{N_L}}{1 - C_{N_L}} \tag{4}$$

Note that it is not necessary to specify the number of CDNs. The CDNs define a partitioning on the universe of objects, but they yield equivalent hit ratios given our assumption that the objects assigned to each CDN show an equivalent popularity distribution. More generally, it can be shown that the hit ratios predicted by the formula in Table 1 are comparable for varying n and λ, as long as n/λ is held constant. Intuitively, partitioning a request stream across any number of caches or CDNs does not change the number of compulsory misses, given the assumption that all requests for any given object are routed to the same cache or CDN.

Moreover, it is not necessary to specify the number of interior caches per CDN, or the replication degree of each object. Our assumption of a fixed N_I parameter fixes the replication degree at N/N_I. The number of interior caches is immaterial, again assuming that n/λ is held constant, i.e., each cache receives requests in proportion to the number of objects assigned to it.

It is interesting to note that since CDNs act as trusted agents of the content providers, they may cache some objects that are not cacheable by leaf caches due to security concerns, reliable hit metering, etc. Thus p_c may have a higher value for interior CDN nodes, yielding higher overall hit ratios for a given cacheable hit ratio.

3 Analytical Results

In this section, we apply the analytical hit ratio model from Section 2 to idealized cache organizations. We vary several key parameters, population, population density, and α, a parameter to the Zipf distribution. We plot the cacheable hit ratio for an interior cache level using parameters similar to those used by Wolman et al.: (Zipf) $\alpha = 0.8$, "popular" Web objects make up 0.3% of all objects, and a slow rate of change for Web objects (averaging one change every 14 days for popular objects, and one every 186 days for unpopular objects).

We also use a per-client request rate λ of 590 requests per day and $n = 3.2$ billion total Web objects. However, as described in Section 2.1, the results apply for other values; it is only the ratio λ/n that is significant. For example, a CDN handles only a subset of all Web objects, and thus sees a lower request rate per client.

The slow rate of change parameter makes the peak hit ratios occur at smaller populations than would a faster rate of change. In this respect, our results are optimistic in projecting the maximum benefit of interior caching by interior population, but conservative in projecting the benefit of interior caches as leaf caches become more effective.

3.1 Interior Hit Ratio

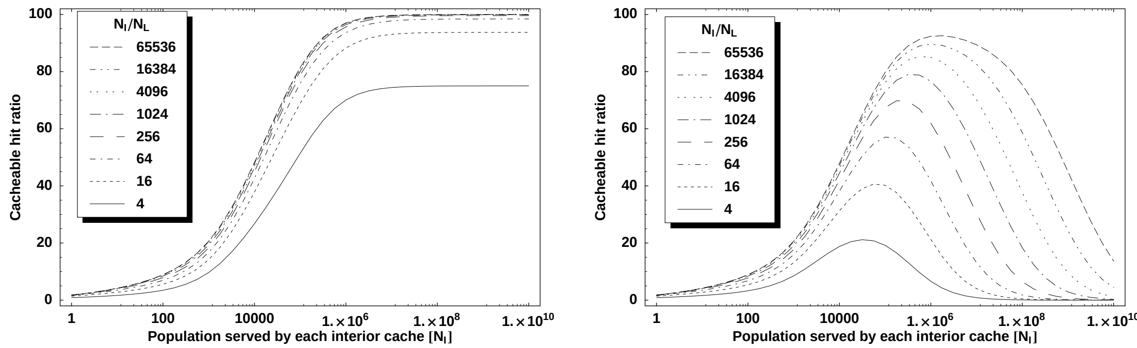
The graphs in Figure 3 define upper bounds on the marginal benefit of interior caching for a population divided into groups of size N_L, each served by one independent leaf cache. The x-axis in Figure 3a/b is the population N_I served by each interior cache; N/N_I is therefore either the number of (independent) root-level caches in a cache hierarchy, or the replication degree in each content-delivery network in a supply-side scenario.

Each plot line in Figure 3a/b represents a different value of N_I/N_L, the number of leaf caches directing some portion of their miss streams to each interior cache. Thus, as we move right on the x-axis, we see how interior caches perform for a given total population partitioned into a a fixed number of client groups (ISPs, for example), each served by a demand-side proxy cache. This type of plot is used in Section 4 to match observed hit rates to an estimate of total population, given a known number of leaf caches.

Figure 3a shows the predicted steady-state cacheable hit ratio *observed* at the interior level. This graph gives a different perspective on the marginal benefit of cache sharing (in this case through an ideal interior cache layer) than presented in the Wolman paper. With N_I ranging from 10K to 1M, an interior cache can absorb a significant share of the miss stream from the leaf-level caches, reducing upstream traffic and load on the origin servers. The interior hit ratio grows rapidly with the interior cache population; with 1M users, it ranges from 65% to 95% of the cacheable miss stream.

Though the cacheable hit ratio in Figure 3a approaches a plateau (nearing 100% when the populations served by each interior cache are divided among a large number of leaf caches), the high hit ratio is misleading at large populations because the actual request rate seen at the interior level also approaches zero. Figure 3b shows the marginal improvement in *overall* cacheable hit ratio offered by an interior cache, i.e., the percentage of *all* cacheable requests that hit at the interior level. As the population N_L served by each leaf cache grows, the marginal benefit of interior caches approaches zero.

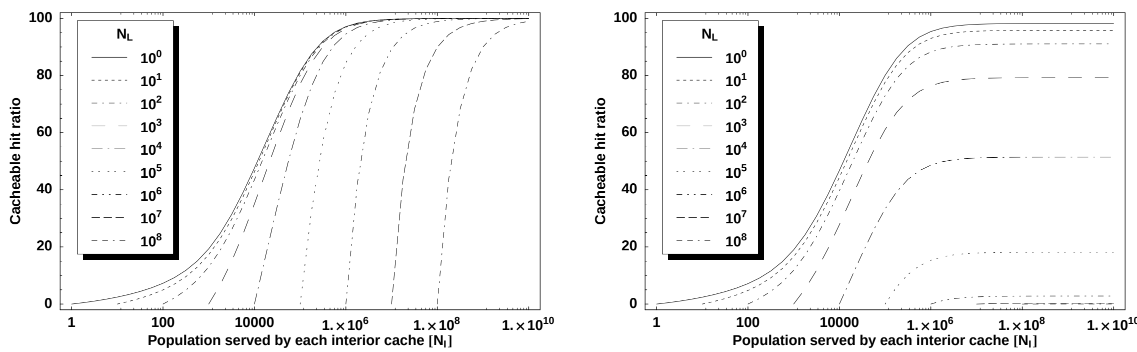
These graphs emphasize a primary conclusion of [15], that the leaf population size N_L is the most important factor determining the marginal benefit of the interior caches as predicted by the model. For example, if a CDN assigned a server to a geographic area whose Web-using population of 10 million is divided among 1000 leaf caches each with ~10K clients (e.g., regional ISPs, universities, businesses), then that CDN server would observe a hit ratio close to 100%, and those hits would account for around 50% of all cacheable requests. However, if the same



(a) observed at interior cache level

(b) as percentage of all cacheable requests

Figure 3: Interior cacheable hit ratios, varying number of leaf-level caches assigned to each interior cache.



(a) observed at interior cache level

(b) as percentage of all cacheable requests

Figure 4: Interior cacheable hit ratios, varying leaf-level cache population density.

client population funneled its request stream through a smaller number of leaf caches (e.g. because of a cache sharing protocol or aggregation of clients into larger groups), then the marginal benefit of the interior caches declines rapidly. In this example, if leaf populations were increased by an order of magnitude, then the interior caches would generate hits for less than 20% of cacheable requests.

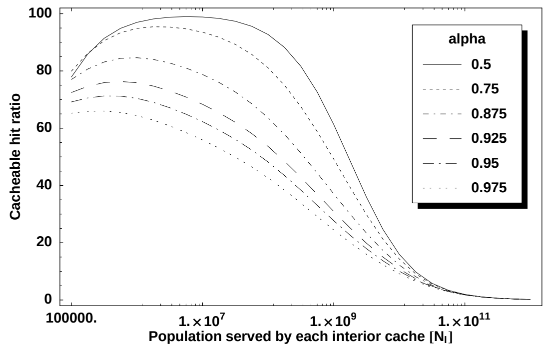
3.2 Interior Hit Ratios with Fixed Leaf Populations

Figure 4 presents the hit ratios of interior caches for fixed population densities N_L at the leaf caches. As the total population increases along the x-axis, the number of leaf caches assigned to each interior cache (N_I/N_L) also increases. This is similar to Figure 3, but moving right along the x-axis shows the effect of scaling population by increasing the number of leaf caches, rather than by increasing their populations.

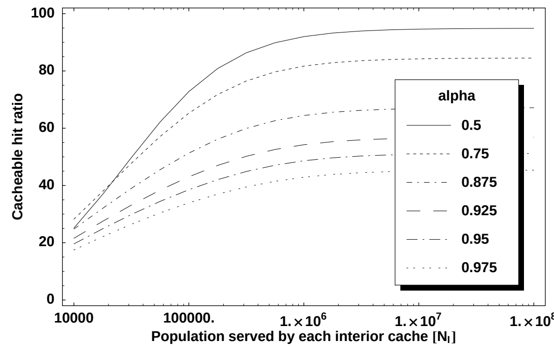
Figure 4a shows the interior cache hit ratio for cacheable requests for selected N_L values as the total population increases. Each line reflects a hit ratio of zero when the interior population matches N_L, i.e., there is only one leaf cache. Hit ratios at the interior caches jump dramatically as a second leaf is added, but level out quickly as further increases in population yield smaller marginal benefits.

Figure 4b shows the marginal benefit to the overall hit rate for all cacheable requests. This graph shows that the the marginal benefit of an interior cache level given fixed N_L is significant up to moderate values of N_L, but decreases rapidly as N_L grows larger.

One important effect is shown by Figure 4b: be-



(a) 65536 leaf caches.



(b) 1024 clients per leaf cache.

Figure 5: Interior cacheable hit ratios, as a percentage of all cacheable requests, varying Zipf parameter α.

yond a certain aggregate population ($N_I = 10^6$ in this case), the marginal benefit of an interior level remains constant *irrespective of the number of leaf caches assigned to each interior cache.* Beyond this point, a caching system or CDN can effectively accommodate larger populations simply by adding independent interior caches; there is no further benefit to increasing N_I.

3.3 Effect of Changing α

In this section, we quantify the effect on interior cache level hit ratios of varying the α parameter to the Zipf distribution. A high α indicates that popular objects are *very* popular, and that unpopular objects are *very* unpopular. As aggregate client request streams become less/more homogenous, it is useful to see if and how particular ranges of α affect the relative contribution of interior or leaf caches to the overall hit ratio. Intuitively, homogenous streams (characterized by a high α) would allow leaf caches to absorb a large number of hits, and non-homogeneous streams (low α) push hit-absorbing responsibilities to interior caches serving larger populations.

The graphs in Figure 5 confirm that changing α has a pronounced effect on the behavior of the interior level. Figure 5a fixes $N_I/N_L = 65536$, and Figure 5b fixes $N_L = 1024$. As expected, Figure 5a shows that small aggregate populations realize a larger percentage of hits when α is large, reducing hits in the interior. However, an interior cache still shows significant benefits when backing either 64K leaf-level caches up to a total population of about 10^8, or leaf caches with small populations, even with values of α as high as 0.975.

4 Correlating the Model With the NLANR Hierarchy

In 1995, the IRCache group at the US National Laboratory for Applied Network Research (NLANR) deployed a group of cooperating Web proxy caches across the United States [14]. Their goal was to provide a large-scale infrastructure for hierarchical Web caching, and to study its behavior. NLANR publishes performance data from its root caches daily, including logs of HTTP requests passing through these caches. The Wolman study implies that the root hit ratio for a large-scale hierarchical cache should be low. Our goal is to leverage the NLANR experiment to validate the analytical model presented in Section 2, by determining if interior hit ratios measured in the NLANR cache match the predictions of the model. We make our observations of the NLANR cache hierarchy using traces collected on October 12, 1999.

The configuration at the end of 1999 included ten large NLANR-operated root caches, each of which acts as a parent for a set of independently operated second-level proxy caches. Since we are concerned only with hit ratios in the roots, we may consider the second-level caches to be leaves. The NLANR roots act as a single unified root cache. Requests are routed/forwarded between the root caches according to a static request routing function that maps the top-level domain (*.edu*, *.com*, etc.) to a fixed subset of root caches. In this respect, the NLANR hierarchy is a precursor to more recent supply-side content distribution services; the primary differences are that the NLANR cache is an agent of the clients rather than of the servers, its routing function is statically configured and maintained by the leaf caches using the Squid proxy cache software, and each interior cache receives the entire miss stream of some subset of the leaf caches. In this case, the routing function increases the population covered by each interior cache with respect to some set of objects, but without decreasing the total size of its covered object universe.

The first step is to determine the aggregate hit ratio for the unified root, which we model as a single interior node. The number of request hits is given directly by the traces. To determine the number of requests to the unified root, we first filter the traces by source IP address to remove requests forwarded between root caches. NLANR anonymizes the IP addresses of all nodes in the traces each day, but it is easy to determine the mapping of root cache names to anonymized IP addresses. This is because each request that is forwarded between root caches shows up at least twice in the traces. The initial request from a leaf cache to a root cache A shows up in A's access trace, with a flag indicating that this request has been forwarded, giving the destination root cache (identified by a fully-qualified domain name, such as *bo1.us.ircache.net*) to which it was forwarded. Given the list of forwarded requests from each root cache A to each root cache B, we can match A to its anonymized IP address in B's access trace.

The observed hit ratio — a respectable 32%, determined by counting all requests labeled as HITs in the original traces (filtered as described above) — is a steady-state hit ratio, meaning that the cache was warm and populated before the start of the measurement period. We also see that for the whole of October 1999, reported hit ratios range between 26% and 36%, but hover close to 32% most days.

Once the unified root hit ratio is known, we apply the analytic model to see if it matches the observations. Unfortunately, several parameters needed to predict hit ratios are not available, and must be inferred from the data. Little data is available from the 914 leaf caches in the hierarchy at that time, e.g., we do not know the size of their user populations or the request rates they serve. This cannot be determined from the root traces because the source of each request is hidden by leaf caches, and hits at the leaves are not recorded in the root traces. We "reverse-engineer" the hierarchy by making reasonable assumptions about typical Web client behavior. The outcome is to use the model and the observed hit ratio to predict the total population served by the NLANR cache in October; this prediction turns out to be on the order of 10,000 clients. If the actual population (which we do not know) diverges significantly from the predictions, then the NLANR experiment may cast doubt on the accuracy of the model or indicate changes in the way it is parameterized.

The accuracy of our prediction also depends on how much the characteristics of the NLANR environment deviate from the architectural assumptions used by the analytical model. For example, the NLANR caches are not unbounded — they may therefore produce capacity misses, a class of misses not generated by the ideal caches of the model. Also, though the NLANR root caches effectively act as a unified root serving an aggregate population of N, side effects of cache routing at the root caches (e.g. redundant copies of objects) may further reduce the hit ratio seen at the actual NLANR roots. We have calculated *stack distances* for these traces, which characterize temporal locality and can predict the amount of storage needed to eliminate capacity misses [2], and the results suggest that the 16GB allocated to each NLANR root proxy is large enough to render these effects negligible.

One last concern arises from the fact that the NLANR caches mark as hits any request whose response *body* comes from local storage. This includes requests that trigger a validation request (e.g. *If-modified-since*) to the home server that result in *Not modified* responses, allowing the cache to return the previously cached object. The analytical model uses the rate-of-change parameter μ to characterize the rate of events that trigger compulsory misses, i.e. either object expiration or object change; in the NLANR caches, however, expired objects are marked as a hit if the cached copy is validated as unchanged by the home server. So, in classifying requests as hits or misses (lower- or higher-cost requests in a sense), if the intended result is to reduce client-side latency,

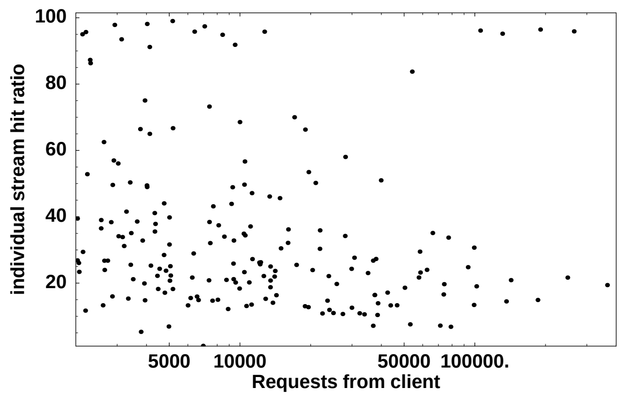


Figure 6: Hit ratio at root for the 200 most active request streams.

one should first determine a reasonable μ that captures the rate at which objects are likely to need validation at the server. However, from a supply-side perspective, validation requests for unchanged objects use less network resources and server CPU cycles than full requests; in the context of reducing these costs, it is therefore appropriate to choose a μ that models the actual rate of change at the server, and to mark the server-validated requests as hits.

4.1 Observations

We first determine the number of "active" leaf caches served by the unified root. Using the filtered trace, we sorted each leaf cache (i.e. each unique requesting IP in each trace) by the total number of requests it sent. We find that less than 200 of the 914 leaf caches account for more than 95% of the requests to the root caches. These 200 "active" caches each sent between 2,000 and 366,000 requests during the 24-hour duration of the trace.

Figure 6 maps, for each active cache, its request rate to its individual hit ratio in the root for the stream of requests it generates. Each dot represents a client. The dots are well-distributed in both axes, indicating that (1) most clients are caches with few customers, and (2) there is little correlation ($\rho(x, y) = 0.022$, or 0.036 including non-active caches) between a cache's request rate and its individual stream hit ratio. This suggests that the NLANR leaf caches exhibit highly diverse workloads, perhaps because small client populations make them susceptible to variations in user behavior.

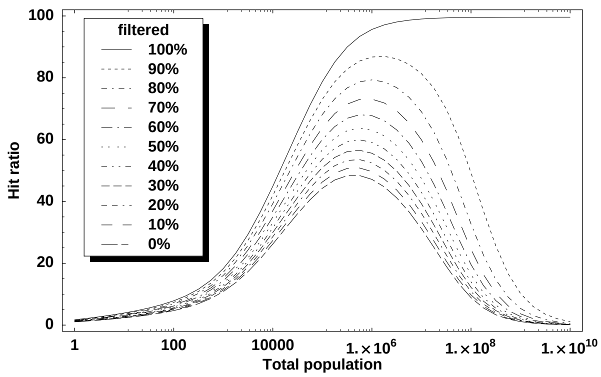
4.2 Correlating with the Model

We begin with our assumptions about the NLANR hierarchy. First, because 200 leaves generate nearly all of the requests, we fix the number of leaf caches at 256 caches. We also assume that 30% of uncacheable documents are immediately recognizable as such (due to clues in the URL, cookies, or expiry) and are not forwarded up the hierarchy to root caches — this number has only a small effect on our predictions, as it turns out below. Lastly, we can estimate the population size from the number of requests in the trace — 7 million. If each user requests an average of 590 requests per day, this indicates about 13,000 users. This estimate is certainly not exact given that many requests are filtered by lower levels, but we can reasonably assume that the population is on the order of tens of thousands.

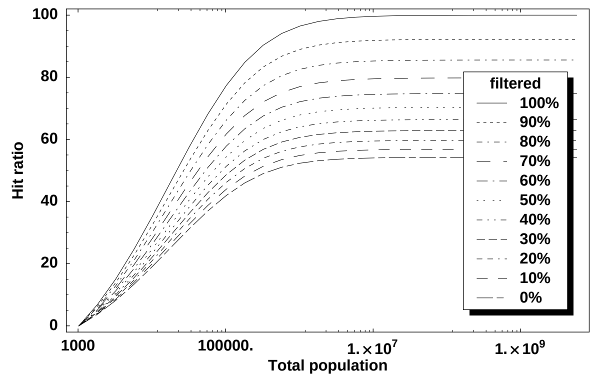
Figure 7a shows the interior hit ratio for 256 leaf caches, varying the percentage p_u of requests detected by leaves as uncacheable. 0% means none are filtered and 100% means all are filtered (corresponding to the line for 256 caches in Figure 3a).

Similarly, Figure 7b shows the interior hit ratio for a client-to-leaf-cache ratio of 1024, varying p_u. Again, 0% means none are filtered and 100% means all are filtered (similar to the 1000 clients-per-cache line in Figure 4a).

Figure 7a indicates that for 256 caches, with a 30% immediate uncacheable detection ratio, and a hit ratio of 32%, we should expect that the number of users of the NLANR cache hierarchy is on the order of 10,000. A population of 20 million would yield the same hit rate, but the daily request rate to the root would be much higher. Note that the left halves of all curves are very close together at this hit ratio, so even if the percentage of uncacheable requests varies from our assumptions, we would still arrive at the same conclusions. Figure 7b also supports our order of magnitude prediction if the leaf caches each had an average of 1024 clients.



(a) 256 leaf caches.



(b) 1024 clients per leaf cache.

Figure 7: Total root hit ratio, varying percentage of uncacheable requests filtered by leaf caches.

5 Conclusion

This paper focuses on deriving hit ratios for *interior* caches in multi-level cache hierarchies, including those incorporating supply-side content distribution networks (CDNs). Evaluating caching configurations by interior hit ratios emphasizes the effectiveness of upstream caches at absorbing the request stream presented to them, in order to further reduce load on the server hosting network and the origin servers. This adds perspective to earlier studies of marginal benefit of upstream caching to overall hit ratios or client-perceived latency.

We show how to apply the analytical model proposed by Wolman et al. recursively to derive interior cache hit ratios in multi-level caches. Using this extension as a starting point, we apply the model to explore the effectiveness of CDNs when demand-side caching is ubiquitous. This scenario is structurally equivalent to a hierarchical demand-side proxy cache in which a request distribution function selects the parent cache (content distributor) for each object requested through the leaf caches.

We present basic results showing the effectiveness of interior caches when demand-side caching is ubiquitous, in the presence of trends that scale either leaf cache populations or the number of leaf caches as the client population increases. We find that if leaf cache populations scale with the client population, interior caches are effective up to interior populations of about 10 million clients. If instead the number of leaf caches increases without increasing the population of each leaf cache, interior cache hit ratios reach a plateau, beyond which there is no benefit to increasing the population served through each interior cache. These results show the implications of the analytical model for CDNs: although CDNs may yield good local hit ratios, they contribute little to the effectiveness of the caching system as a whole when leaf populations are large. Given that leaf populations will grow as ISPs consolidate and serve larger user populations, this suggests a natural limit to the benefits of CDNs given recent empirical observations of Web traffic. Although we make various simplifying assumptions to apply the model to CDNs, the basic conclusions of the model are inevitable.

However, these conclusions are valid only if the parameters and assumptions of the analytical model match current Web traffic patterns. For example, CDNs are more beneficial when the object universe includes significant numbers of popular objects that are very large or that change rapidly. To help validate and parameterize the model, we apply the interior cache hit formulas to show how to correlate the Wolman model with real-world performance data from the NLANR cache hierarchy. Our approach yields predictions for the population served by the NLANR cache given performance data from the NLANR roots. If additional empirical data about the NLANR user population becomes available, this technique could further validate the model or adjust parameters as Web access patterns evolve.

References

[1] Akamai Technologies, Inc. FreeFlow overview. http://www.akamai.com/service/freeflow.html.

[2] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliviera. Characterizing reference locality in the WWW. In *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pages 92–103, December 1996.

[3] Paul Barford and Mark E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of Performance '98/ACM SIGMETRICS '98*, pages 151–160, June 1998.

[4] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of IEEE Infocom '99*, March 1999.

[5] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical Internet object cache. In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.

[6] Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. Summary Cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of SIGCOMM '98*, September 1998.

[7] Syam Gadde, Jeff Chase, and Michael Rabinovich. Directory structures for scalable Internet caches. Technical Report CS-1997-18, Department of Computer Science, Duke University, November 1997.

[8] P. Krishnan and Binay Sugla. Utility of cooperating Web proxy caches. In *Proceedings of the 7th International World Wide Web Conference*, April 1997.

[9] Ingrid Melve. When to kill your siblings: cache mesh relation analysis. *Computer Networks and ISDN Systems*, 30(22-23):2105–2111, 1998.

[10] Mirror Image Internet, Inc. instaDelivery Internet services. http://www.mirrorimage.com/.

[11] Sandpiper Networks/Digital Island, Inc. Footprint content delivery services. http://www.digisle.net/services/cd/.

[12] Vinod Valloppillil and Keith W. Ross. Cache Array Routing Protocol v1.0. Internet-Draft, June 1997.

[13] Duane Wessels et al. Squid Internet Object Cache. http://squid.nlanr.net/.

[14] Duane Wessels, Traice Monk, k claffy, and Hans-Werner Braun. A distributed testbed for national information provisioning. http://ircache.nlanr.net/Cache/.

[15] Alec Wolman, Geoff Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry Levy. On the scale and performance of cooperative web proxy caching. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, December 1999.