

Uncacheable Documents and Cold Starts in Web Proxy Cache Simulations: How Two Wrongs Appear Right

Sandra G. Dykes, Kay A. Robbins
Division of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249-0667
{sdykes, krobbins}@cs.utsa.edu

Clinton L. Jeffery
Department of Computer Science
University of Nevada at Las Vegas
Las Vegas, NV 89120
jeffery@cs.unlv.edu

Abstract—This study examines a series of 16 single day traces from two NLANR proxy caches. We find clients often repeat their requests for an uncacheable document, forcing the cache to re-fetch the document multiple times. This behavior is primarily caused by HTTP cache control headers rather than dynamic content. Assuming all documents are cacheable falsely predicts cache *hits* for 20% to 25% of the requests. Starting a simulation with a cold cache falsely predicts cache *misses* for approximately 20% of the requests. With both, the errors offset and the predicted hit rate appears to match the measured hit rate, possibly leading to acceptance of a poor model and incorrect simulation results for other workloads. Repeat requests from the same client also inflate Zipf parameters and increase their variability. Without them, the median α falls from 0.8 to 0.3, and the median Ω falls from 3500 to near 30. Corrected Zipf parameters provide better insight into cache sharing patterns, uncover periodic behavior, and more accurately parameterize hit rates models.

Keywords— Web caching, proxy caching, cooperative caching, Zipf, trace-driven simulation, NLANR, Squid, repeat requests

I. INTRODUCTION

Web caches exist at several levels: browsers maintain a private cache for the user (L1), organizational proxy caches share documents among their users (L2), and cooperative proxy caches share documents between proxies (L3). Although researchers agree that user and proxy caching improve Web performance [1][2], there is considerable debate about the fundamental structure of cooperative Web caches [3]. Recently Wolman et al. have questioned whether *any* form of cooperative Web caching provides significant benefit over L2 proxy caching [4].

Definitive answers to the viability and performance of cooperative Web caches are difficult to obtain because data must contain simultaneous request streams from multiple, independent proxy caches. Consequently, most Web caching research relies on either trace-driven simulation or analytical models whose parameters are derived from proxy traces. One important source of proxy traces is the National Laboratory for Applied Network Research (NLANR), which operates a global cache hierarchy using Squid proxy caches [5]. NLANR traces are widely used in Web caching research because they are publicly available, up-to-date, and include many diverse users [6][7][8][9].

This paper points out two pitfalls in trace-dependent Web cache modeling and shows how their errors can offset, resulting in an apparent match between predicted and measured hit rates. Such a match could lead to the acceptance of an inaccurate model and possibly result in misleading predictions.

This work was supported by National Science Foundation grant CDA-9633299.

First, most duplicate requests in the NLANR proxy traces are repeat requests from the same client, and 90% to 95% of these are cache misses. In one trace, the most popular document received 4920 requests, all from the same client. All were cache misses, despite the fact that the document was successfully retrieved each time, was neither a cgi-bin nor a query, and the requests did not attach a cache control header. We later determined that response headers from the origin server were responsible. After removing repeat requests from the same client, the most popular document received only 34 requests. Simulations driven by most proxy traces have difficulty accurately determining cacheability because standard trace formats do not record response header information. Our results show that assuming all documents are cacheable predicts false hits for approximately 20% to 25% of the requests, determining cacheability from information in standard traces predicts false hits for 10% to 20% of the requests, and that inferring uncacheability from repeat misses corrects this problem.

A second error occurs if the simulation starts with a cold cache. For 1-day NLANR traces, starting with a cold simulation cache predicts false misses for approximately 20% of the requests. This leads to an interesting interplay between cacheability and warmup errors. As the errors are of similar magnitude but opposite direction, ignoring both may predict hit rates that match measured hit rates and appear to validate the simulation model. In fact, the model would be inaccurate and could predict erroneous results for other workloads.

Repeat requests also inflate the Zipf parameters. Zipf's law is a power-law function that relates the frequency of an event to its popularity rank. In Web caching, Zipf's law relates number of requests for a document to its popularity rank, and has been interpreted as a measure of potential cache sharing and ideal hit rate. *Our results show the Zipf exponent appears large because there are many uncacheable repeat requests from the same client, not because there is potential document sharing.* If all documents were cacheable, repeat requests would disappear because Web browsers would cache the documents and the proxy cache would not see the requests. Consequently, Zipf distributions that include repeat requests do not reflect actual or potential cache sharing, either across clients or across all requests. We show that removing repeat requests yields much lower Zipf parameters that reveal caching patterns and better reflect sharing across clients.

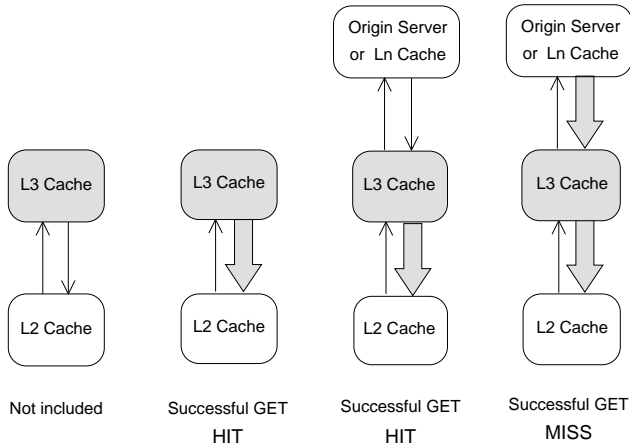


Fig. 1. Web cache scenarios illustrating the definition of hit rate. Narrow arrows represent a HTTP header; wide arrows represent the transfer of a document. The leftmost scenario is not a successful GET and is not included in the analysis. The middle two scenarios are L3 cache hits, and the rightmost is an L3 cache miss.

Section II discusses background issues, including a definition of hit rate for Web caches, example cases of repeat requests, and a more in-depth discussion of Zipf parameters. Related work in Web cache simulation and modeling is described in Section III. Section IV explains our methodology, and Sections V and VI contain results and conclusions.

II. ISSUES IN WEB CACHE MODELING

A. What is a cache hit?

Results from different Web caching studies are difficult to compare because there is no standard definition for cache hit rate. Some researchers report results based upon number of successful GET requests [6][10], while others base it upon all requests, including those with errors [4]. This distinction is critical because unsuccessful requests are common: on the average, we found successful GETs comprise only slightly over half of all requests in the NLANR traces.

Likewise, there is no agreement on the definition of a proxy cache hit. Figure 1 illustrates the four possible scenarios:

1. the proxy does not return a document because the request is not a successful GET
2. the proxy returns the cached document without validation
3. the proxy returns the cached document after exchanging HTTP validation headers with the origin server
4. the proxy retrieves the document from the origin server and forwards it to the client.

Scenario 3 is neither a cache hit nor a miss in the traditional sense; however, in both response time and resource usage it is more comparable to a cache hit because HTTP headers are far smaller than most documents and the file I/O is not involved. We therefore define hit rate based upon document transfer. Hereafter, the term “request” should be taken to mean a successful GET request, and cache hits are defined as in Figure 1.



Fig. 2. A grey GIF image frequently retrieved by the NLANR caches.

B. Repeat requests and Web advertisers

We define a repeat request as a request from the same client for the same URL, and observe that, in the NLANR traces, 90% to 95% of repeat requests are cache misses. Web advertisers supply some of the more egregious examples of such requests. Figure 2 displays the document returned most often from the NLANR BO1 cache on January 30: a 1×1 grey GIF background from a Web advertiser. The same grey GIF image was also the 2nd and 5th most frequently returned document. The proxy cache returned these three documents almost 20,000 times, yet only 23 of the requests were cache hits. Equally as striking, these thousands of retrievals were for at most 26 clients. The most popular document was retrieved 14,210 times for only 4 clients, and no document was returned to over 16 clients. We noticed other advertisers use an identical grey GIF image. In a different trace, the cache fetched this image from RealMedia a total of 2730 times, returning it each time to the same client.

Table I lists the five most popular documents in the January 30th trace, together with the number of requests, cache hits, and clients, and shows a pattern of repeat misses that is typical of all traces in this study. Throughout the traces, we observe repeat requests often access Web advertisement sites. Of the five most popular documents in the January 30th trace, four are to a DoubleClick site and the fifth to Netscape.

It is interesting to examine why a small static image is uncacheable. The log file indicates that requests for two of the documents contain cache control directives, but does not explain why the third is uncacheable. To test if response headers are responsible, we retrieved the documents with a custom `httpget` utility. All three origin servers attach `ETag` headers with entity tags that change every minute or two, despite the fact the document is unmodified. Entity tags are designed to assure cache consistency: the server attaches an entity tag to the document and the client includes this tag with subsequent requests. If the document is unmodified, the server should respond with an HTTP 304 Not Modified message instead of the object. Our tests, however, show the servers improperly resend the document and thereby generate cache misses. Even if origin servers respond correctly, it remains unclear why a static 1×1 GIF image should require consistency checking on every request.

TABLE I
FIVE MOST POPULAR DOCUMENTS IN BO1 TRACE, JAN. 30, 2000.

Req.	Hits	Clients	Bytes	URL
14210	0	4	409	m.doubleclick.net/viewad/817-grey.gif
4791	2	6	244	ad.doubleclick.net/viewad/817-grey.gif
1394	0	1	12821	m.doubleclick.net/viewad/385809-family. . .
939	0	1	2130	m.doubleclick.net/viewad/28902-lycoshop. . .
915	21	16	359	messenger.netscape.com/images/pixel.gif

C. Zipf parameters

A Zipf distribution relates number of requests for an object, R , to its popularity rank i :

$$R(i) = \frac{\Omega}{i^\alpha}.$$

The exponent α reflects the popularity skew, and the constant Ω approximates the number of requests for the most popular document ($i = 1$). Because α increases as popular documents receive a greater fraction of requests, it has been interpreted as a measure of the potential sharing in a Web cache. An α near one suggests a strong potential for document sharing and a high cache hit rate.

Beginning with Glassman in 1994 [11], numerous studies report Web requests follow a Zipf or Zipf-like distribution [6][12][13][14][15], with α ranging from 0.64 to 1.0. However, all previous Zipf’s law studies in Web caching that we are aware of include repeat requests. The analysis in Section V-E shows *Zipf’s law behavior and large α ’s are caused by repeat requests for uncacheable documents, not by potential sharing between clients.*

When repeat requests are removed, α drops to 0.2 – 0.3. The lower α ’s are a truer indication of the potential cache sharing and hit rates. Further, if changes to Web applications or protocols remove some cacheability constraints, the associated repeat requests will disappear because lower level browser caches or proxies will store the document and clients will not need to repeat their requests. Under no circumstances do repeat requests reflect document sharing, and α ’s computed with them misrepresent potential cache sharing and hit rates.

III. RELATED WORK

Cao and Irani [16] found a large number of repeat requests in traces from DEC, University of Virginia, and Boston University. Their data show users often re-access the same document, and that the fraction of repeat requests peaks on a 24 hour cycle. Wolman et al. report that 40% of all requests to shared documents at the University of Washington are repeats by the same client [17], and Nishikawa et al. [14] observe that users tend to access the same document repeatedly at short intervals. Nishikawa hypothesizes that removing such requests from the trace might improve their simulation results.

Wolman et al. [4] collected traces that include response header information by installing custom monitoring software at the University of Washington’s Internet gateway. The authors

also obtained concurrent traces from proxies at Microsoft Corporation. Because they controlled data collection, the authors could avoid cacheability and warmup errors; however, this approach is limited to sites that grant access. Wolman extrapolates the University of Washington trace to multiple sites by creating virtual proxies according to client IP address. In this study, all users were located in the same geographic area (Seattle) and either attended the same university or worked for the same company, limiting the diversity of the user population.

Breslau et al. [6] take the opposite approach, analyzing traces from six large proxy caches that include L3 caches at NLANR and Questnet in Australia, and L2 caches in Italy, Finland, and the United States. This approach ensures diversity and measures actual cache behavior, but allows no control over the logged information. For the NLANR trace, Breslau includes successful GET requests for both cacheable and uncacheable documents in the analysis. The authors report Zipf parameters of $\alpha = 0.64$ and $\Omega \approx 4000$. As NLANR cache sites typically service less than 100 clients and the authors used four NLANR sites, we estimate the number of clients to be less than 400. If there were no repeat requests, Ω would not exceed the number of clients, yet in Breslau’s data it is greater by a factor of ten. Breslau’s study is the most comprehensive to date on the relationship of Zipf’s law to Web caching and provides important information; however, the inclusion of repeat requests adversely affects hit rate predictions that depend upon the Zipf parameters.

Duska et al. [7] examine inter-trace sharing across six proxy caches and report results as percentage of shared URLs rather than as Zipf distributions. Using their reported data, we determined Zipf parameters for the combined traces. Assuming no repeat requests, and no capacity or consistency misses, we computed least squares fit parameters of $\alpha = 0.24$ and $\Omega = 10$. These parameter values are far lower than those reported in other studies, and match results reported here.

IV. METHODOLOGY

A. Traces

This study uses publicly available traces [5] from the NLANR Boulder (BO1) and Urbana-Champagne (UC) caches for the 16 days from January 27 to February 11, 2000. These caches are hierarchical L3 caches which service only L2 proxies; approximately 90 proxies use BO1 as the parent cache, and approximately 60 use UC. Client IP addresses are randomized daily, so they are consistent within a trace but not between traces. Each trace spans one day and contains from 400,000 to 900,000 total requests for a combined total of 9 million requests for BO1 and 11 million for UC. Log files use the Squid native log format, which records the cache action and allows computation of actual hit rates.

B. What is cacheable?

We determine cacheability from explicitly logged information and from cache behavior. A request is considered uncacheable for the following reasons:

- *Cgi-bin (C)* - The URL contains “cgi-bin”.

- *Query (Q)* - The URL contains “?”.
- *Pragma (P)* - The client issued the request with a no-cache pragma or analogous cache control command (Squid code TCP_CLIENT_REFRESH_MISS).
- *Partial Content* - Request is for partial content (Status 206).
- *Inferred (X)* - None of the above apply, yet the document receives ≥ 2 requests and all are logged as cache misses. We attribute X uncacheability to response headers for cache control and to less obvious mechanisms such as frequently changing entity tags.

C. Cacheability models

Four different models are used to measure the effects of cacheability assumptions and of warming the simulation cache. All treat the initial request for a document as a cache miss. Duplicate requests for cacheable documents count as cache hit, and duplicate requests for uncacheable documents count as cache misses. Models assume infinite cache size, and, with the exception of the Recorded model, assume no consistency misses. Results show these assumptions introduce little error in the computed hit rates. The four cacheability models are:

All requests: All documents are considered cacheable.

Cacheable CQP: Cgi-bin, query, pragma, and requests for partial content are considered uncacheable.

Cacheable CQP-X: In addition to CQP, any document which receives ≥ 2 requests without a cache hit is considered uncacheable.

Recorded: Duplicate requests are assigned the cache action recorded in the log file. In the warmup experiment, the Recorded hit rate will eventually equal the measured hit rate if no documents were replaced by the physical cache.

V. RESULTS

Table II contains per-trace statistics for the two cache sites, and shows the fraction of successful GETs that are initial requests for an object, duplicate requests from different clients, and repeat requests from the same client. Approximately half of all requests are successful GETs: 49% for BO1 and 57% for UC. At BO1, 28% of the successful GET requests are uncacheable. Of these, 6% are initial requests, 2% are duplicate requests from different clients, and 20% are repeat requests from the same client. The ratio of initial requests to repeat requests shows that, on the average, *clients request an uncacheable document not once, but 4.3 times*. The proxy cache must re-fetch the document in over 90% of these cases. The UC cache averages a slightly higher ratio of 5.5 requests and over 5 fetches per client for each uncacheable document.

TABLE II
TRACE STATISTICS FOR NLANR CACHES, JAN 27 - FEB 11, 2000.
FRACTIONS ARE BASED ON SUCCESSFUL GET REQUESTS AND VALUES ARE AVERAGES UNLESS OTHERWISE NOTED.

	BO1	UC
Traces	16	16
Length of each trace	1 day	1 day
All requests per trace	563956	691770
Successful GET requests per trace	276481	397363
Documents per trace	198363	290923
Cacheable	0.72	0.76
Initial	0.65	0.70
Duplicate	0.08	0.07
Uncacheable	0.28	0.24
Initial	0.06	0.04
Duplicate (different clients)	0.02	0.01
Repeat (same client)	0.20	0.18
File size, median		
Cacheable	3.0 KB	3.3 KB
Uncacheable	2.6 KB	2.8 KB

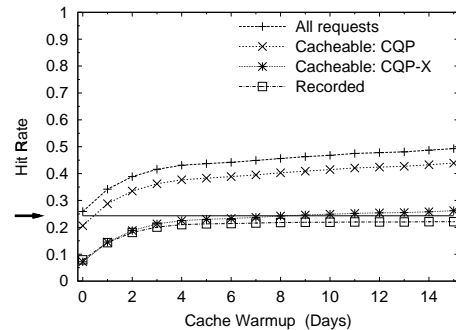
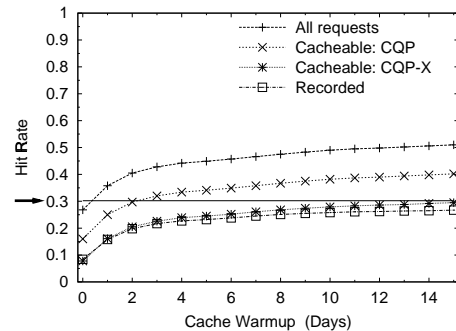


Fig. 3. Effect of cache warmup on hit rates at BO1 (top) and UC (bottom) for the various cacheability models. Hit rates are for the Feb 11 trace, with warmup from Jan 27 - Feb 10.

TABLE III

HIT RATES FOR COLD AND WARM SIMULATION CACHES FOR THE FEB 11 TRACE, WITH CACHE WARMED USING JAN 27 - FEB 10 TRACES (15 DAYS).

	B01		UC	
	Cold	Warm	Cold	Warm
Computed hit rates				
All_requests	0.27	0.51	0.26	0.49
Cacheable CQP	0.16	0.40	0.21	0.44
Cacheable CQP-X	0.08	0.30	0.07	0.26
Recorded	0.08	0.27	0.08	0.22
Measured hit rate	0.30		0.24	

A. Cache warmup

The cache warmup experiment begins with an empty simulation cache. Hit rates are computed for the February 11 trace using each cacheability model. Next, documents from the preceding day's trace are added to the simulation cache, and the February 11 hit rates are recomputed. This process continues until, at the final data point, the simulation cache contains all documents successfully returned by the proxy during the 15 days from January 27 to February 10. At each data point, only requests in the February 11 trace are used to compute hit rates; requests from earlier traces serve merely to warm the cache.

Figure 3 and Table III present cache warmup results for the four cacheability models. Measured hit rates of 0.30 for BO1 and 0.24 for UC are denoted by solid lines in the graphs. As Figure 3 illustrates, the Recorded hit rates closely approach, but never quite equal, the measured rates. This slight difference suggests some cache replacement occurs over the 16 day period, but the amount is small and has little effect on hit rates.

From the warmup graphs we can determine the number of days required to warm the cache. For UC, warmup requires around 4 days, while BO1 requires 8 to 10 days. After this period the recorded hit rates level off and additional warmup is unnecessary. Actual warmup length may differ for other days or other caches; however, these results indicate it is on the order of several days and that simulations driven by single day traces cannot ignore it.

B. Hit rates for different cacheability assumptions

For the ideal cacheability model, the predicted hit rates greatly overestimate the actual hit rates. With a warm cache, the All_Requests model predicts hit rates of 0.51 for BO1 and 0.49 for UC, as compared to the measured rates of 0.30 and 0.24. This overestimate is consistent: assuming ideal cacheability falsely predicts cache hits for approximately 20% of the requests throughout the warmup period.

Large errors for CQP demonstrate that request data is insufficient for modeling cacheability. With a warm cache, the CQP model predicts hit rates of 0.40 for BO1 and 0.44 for UC. Thus, using explicit information in the log file for document cacheability corrects only half the errors in BO1 hit rates, and a fourth the errors in UC hit rates. Simulations must model cache misses due to the actions of HTTP response headers, even

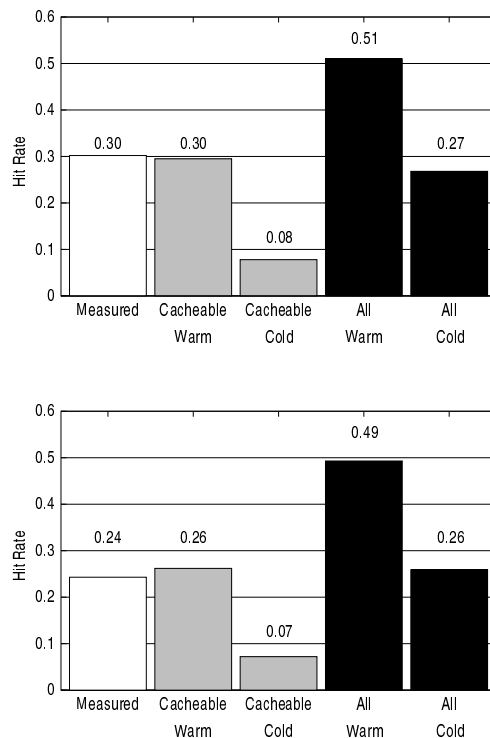


Fig. 4. Measured and calculated hit rates for BO1 (top) and UC (bottom).

though this information is not recorded in most publicly available logs.

As Figure 3 illustrates, the CQP-X hit rates closely overlap the Recorded hit rates throughout the warmup period, and, after adequate warmup, correctly predict the measured hit rate. The agreement establishes that inferring uncacheability from repeat requests can produce a correct model, even if log files do not contain response header information. Our approach makes publicly available traces more useful and reduces the need for collecting custom traces.

C. How cacheability and warmup errors may cancel

Our analysis highlights an unexpected result: when a simulation makes the dual errors of starting with a cold cache and assuming all documents are cacheable, the predicted hit rate approximately equals the measured hit rate. Figure 4 illustrates the separate and combined effects of these two errors. Correctly modeling uncacheability but using a cold cache *underestimates* the hit rate by 22% for BO1 and 21% for UC. Warming the cache but treating all documents as cacheable *overestimates* the hit rate by 17% and 25%. Applying both corrections produces hit rates that closely match the measured rates. Unfortunately, the two errors are of similar magnitude and opposite sign, therefore applying *neither* correction also produces hit rates that closely match the measured rates. This could lead researchers to believe that a simulation model is correct, when in fact these two errors will not necessarily cancel when the simulation workload is changed.

TABLE IV
UNCACHEABILITY ANALYSIS: FRACTION OF SUCCESSFUL GETS
REQUESTS IN NLANR TRACES, FEB 11, 2000.

	BO1	UC
Cacheable	0.75	0.76
Uncacheable	0.25	0.24
206: Partial Content	0.00	0.00
Cgi-bin (C)	0.01	0.00
Query (Q)	0.08	0.01
Request pragma or directive (P)	0.07	0.07
Response header (X)	0.10	0.16
Repeat requests	0.17	0.17
206, Cgi-bin, Query, or Pragma (CQP)	0.09	0.04
Response header (X)	0.08	0.12

D. HTTP cache control and static documents

Repeat misses occur when the document is uncacheable, as in a cgi-bin or query response, or when caching is prevented by HTTP headers. Some headers prevent the cache from storing the object, while others force the cache to validate and possibly re-fetch a document before reusing it. Table IV gives reasons for uncacheability and the corresponding fraction of requests. Values are reported for the Feb. 11 traces and are representative of the other traces in the study. The data show requests for partial content and cgi-bin documents are negligible. Uncacheability is primarily due to HTTP header cache control, which appear in around 17% of the requests. Cache control headers are attached primarily to static objects rather than to queries or cgi-bin documents. Recalling the example grey GIF image, it appears HTTP cache control mechanisms are being used to prevent caches from sharing static documents, even across requests from the same client.

E. Repeat requests and Zipf parameters

Figure 5 shows log-log Zipf plots for the February 7th BO1 trace, and is representative of Zipf plots for the other traces. One plot includes all requests and the other shows the Zipf distribution without repeat requests. Values for α and Ω are determined by a nonlinear least-squares (NLLS) Marquardt-Levenberg algorithm, and the best-fit function is superimposed upon the trace data. Because requests are discrete and the data are dominated by points with 1 or 2 requests, each point in the NLLS is weighted by $1/d$, where d is the number of documents with the same request count. This is equivalent to using the average document rank for each request value, thereby avoiding an artificial flattening of the fitted curve due to the stair-step nature of the data.

Table V gives medians and ranges of Zipf parameters for the 16 daily traces for each cache. Two features stand out: 1) repeat requests greatly inflate both α and Ω , and 2) repeat requests create large variations in the parameters. With repeat requests, the NLANR α ranges from 0.66 to 1.01 for BO1, with a median of 0.83, and ranges from 0.70 to 0.92 for UC, with a median of 0.80. The values for α in these 32 traces overlap the range of previously reported α 's for

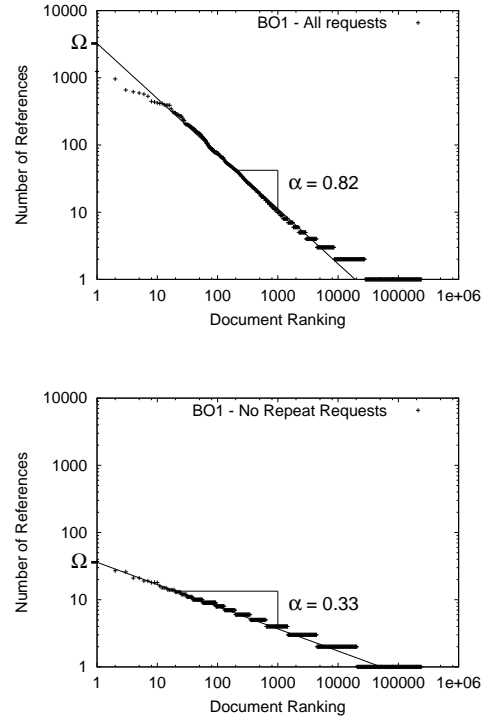


Fig. 5. Zipf distributions for all requests (top) and without repeat requests (bottom), NLANR BO1 cache, Feb 7, 2000.

TABLE V
ZIPF PARAMETERS FOR NLANR TRACES, JAN 27 TO FEB 11, 2000.

	BO1		UC	
	α median	α range	α median	α range
All requests	0.83	0.66 - 1.01	0.80	0.70 - 0.92
No repeat requests	0.33	0.29 - 0.46	0.22	0.19 - 0.31
Mon - Fri	0.31	0.29 - 0.35	0.21	0.19 - 0.25
Sat, Sun	0.42	0.38 - 0.46	0.29	0.27 - 0.31
	Ω median	Ω range	Ω median	Ω range
All requests	3120	1250 - 8700	3500	1450 - 6240
No repeat requests	33	31 - 38	21	18 - 23

NLANR traces [6][12][11][13][14][15]. The high variability across traces is due to random factors in repeat requests and explains the spread in previously reported values. When repeat requests are removed from the traces, the median α falls dramatically to 0.33 for BO1 and 0.22 for UC. Moreover, the ranges tighten to where the α for BO1 is significantly greater than the α for UC, showing that BO1 clients share more documents than do UC clients.

Eliminating the noise of repeat requests also uncovers an interesting sharing pattern in weekdays vs. weekends: α 's for M-F are clearly smaller than the weekend α 's, suggesting there is less diversity during weekends than during the week.

Zipf parameters computed without repeat requests offer better insights into the underlying data. For example, BO1 traces contain around 90 clients, and UC traces around 65 clients. An Ω of 3120 has little meaning because it is the sum of pathologi-

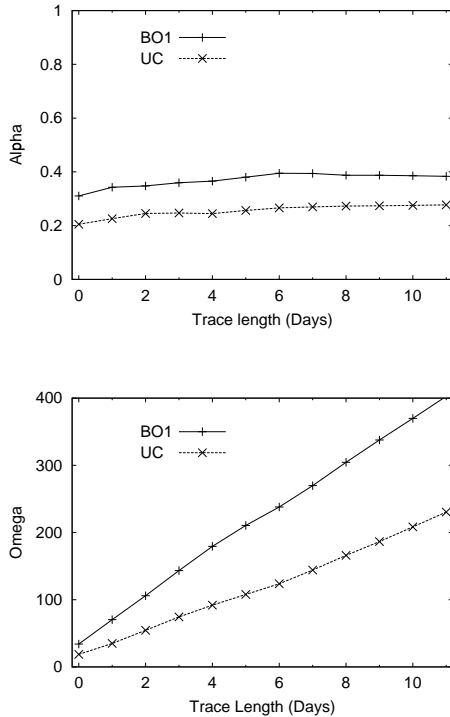


Fig. 6. Effect of trace length on Zipf parameters α (top) and Ω (bottom). Repeat requests have been removed from traces.

cal events. In contrast, the value of Ω computed without repeat request indicates the most popular document is typically shared by 33 different clients, or roughly one-third of BO1’s population. Likewise, the $\Omega = 21$ for UC shows its most popular document is shared by about one-third of its 65 clients.

We demonstrate how Zipf parameters vary with trace length by iteratively merging a preceding day’s trace file and computing α and Ω . Repeat requests are removed from all traces. Results in Figure 6 show α remains fairly constant as the trace length increases, while Ω grows in a linear fashion with slope of approximately 36 for BO1 and 18 for UC. An intuitive interpretation lies in the observation that combining two Zipf distributions with the same α and Ω produces a Zipf distribution with the same α , but with an Ω twice as large. Because Zipf parameters do not vary much from day to day, adding a day’s trace has little effect on α but adds a constant amount to Ω .

VI. CONCLUSIONS

Exploring repeat requests and cache warmup effects uncovers several interesting results with implications for Web cache modeling and practical applications, including the following:

- **Repeat requests artificially inflate ideal cache hit rates.**

An ideal cache assumes all documents are cacheable, and treats repeat requests as cache hits. Many researchers use the ideal cache hit rate as an upper bound for proxy and cooperative Web caches. The difference between ideal and actual hit rates is primarily due to repeat requests because they constitute a substantial fraction of the requests and over 90% are actually cache

misses. In this study, the percentage of repeat requests is 17%, compared to a difference of 21% to 26% between ideal and measured hit rates. *However, if all documents were cacheable, most repeat requests would disappear because of caching by the Web browser and L2 proxies, therefore an ideal hit rate computed using repeat requests is meaningless.*

- **Repeat requests artificially inflate Zipf parameters.**

Zipf parameters are inflated by repeat requests for the same reasons as the ideal hit rate, and do not accurately portray the potential proxy cache sharing between clients or requests. After removing repeat requests, the Zipf exponent, α , drops from around 0.8 to 0.2 for the UC cache and to 0.3 for the BO1 cache. The constant Ω drops from around 3500 to near 30. These corrected parameters offer precise interpretations: α reflects the duplication in requests from different clients, and Ω reflects the number of clients that share the most popular document. Parameter variability also drops sharply, exposing a difference in α between the UC and BO1 cache sites, and between weekdays and weekends. With repeat requests, these differences are not visible due to noisy data.

- **Simulation caches must be adequately warmed.**

We found 4 days was sufficient for one cache, but the other cache required over a week. With 1 day traces, a cold simulation cache predicts false misses for around 20% of the requests.

- **Web cache simulations must correctly model document cacheability.**

Assuming all documents are cacheable predicts false hits for 20% to 25% of the requests. Determining cacheability from only request data and ignoring response headers predicts false hits for 10% to 20% of the requests.

- **Uncacheability can be inferred using repeat misses.**

If a trace does not contain explicitly recorded information for response headers, it can be corrected using repeat misses to infer if a document is cacheable. We would, of course, prefer Squid and other popular proxy cache software include this information in their standard log format.

- **Publicly available traces are essential, but researchers should beware of the cacheability and warmup pitfalls.**

NLANR offers Web researchers an invaluable resource by providing up-to-date, publicly available traces of a large operational proxy cache. These traces are not just a convenience, they are a necessity because most researchers do not have the access required to collect comparable traces. However, researchers should be aware that NLANR and most other Squid traces need to be corrected for response header and cache warmup effects. These two errors may cancel for the test workload, but that does not assure they will cancel for other simulation workloads. Ignoring these pitfalls may therefore result in erroneous simulation results.

- **HTTP headers are being used and misused to prevent cache sharing of static documents**

In addition to cache control directives, some content providers appear to be using HTTP validation mechanisms such as entity tags to prevent caches from sharing static documents, even between requests from the same client.

- **Can repeat requests be eliminated from proxy caching?**

Yes - Web browsers and proxy caches should notice when a document is not being cached, and send subsequent requests directly to the origin server. This would reduce the load on upper cache levels by nearly 20% and remove excess network traffic by eliminating useless document transfers through the cache hierarchy. These are relatively minor modifications to current proxy caching software which could have a substantial impact on performance, and would have the side effect of producing traces that more clearly reflect Web sharing potential.

REFERENCES

- [1] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching proxies: Limitations and potentials," in *Proc. of the 4th Int'l. World-Wide Web Conf.*, Dec. 1995, pp. 119–133.
- [2] J. E. Pitkow, "Summary of WWW characterizations," in *Proc. of the 7th WWW Conf.*, Brisbane, Australia, Apr. 1998.
- [3] S. G. Dykes, C. L. Jeffery, and S. Das, "Taxonomy and design analysis for distributed web caching," in *Proc. of the IEEE Hawaii Int'l. Conf. on System Sciences (HICSS'99)*, Maui, HI, Jan. 1999.
- [4] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative Web proxy caching," in *Proc. of the 17th ACM Symp. on Operating Systems Principles*, Dec. 1999.
- [5] National Laboratory for Applied Network Research (NLANR), "Ircache project," <http://ircache.nlanr.net/>.
- [6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. of IEEE Infocom '99*, Mar. 1999.
- [7] B. M. Duska, D. Marwood, and M. J. Feeley, "The measured access characteristic of World-Wide-Web client proxy caches," in *Proc. of the 1997 USENIX Symp. on Internet Technology and Systems (USITS'97)*, Dec. 1997.
- [8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol," in *ACM SIGCOMM '98*, Vancouver, Canada, 1998.
- [9] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based caching for Web servers," in *Proc. of the SPIE/ACM Conf. on Multimedia Computing and Networking (MMCN)*, San Jose, CA, Jan. 1998.
- [10] M. F. Arlitt and C. L. Williamson, "Web server workload characterization: The search for invariants," in *Proc. of ACM SIGMETRICS*, May 1996.
- [11] S. Glassman, "A caching relay for the World Wide Web," in *Proc. of the First Int'l. World Wide Web Conf.*, May 1994, pp. 69–76.
- [12] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," in *Proc. of the IEEE Conf. on Parallel and Distributed Systems (PDIS'96)*, Dec. 1996.
- [13] C. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW client-based traces," Tech. Rep. TR-95-010, Computer Science Dept., Boston University, Boston, MA, 1995.
- [14] N. Nishikawa, T. Hosokawa, Y. Mori, K. Yoshida, and H. Tsuji, "Memory-based architecture for distributed WWW caching proxy," in *Proc. of the 7th WWW Conf.*, Apr. 1997.
- [15] W. Li, "References on Zipf's law," <http://linkage.rockefeller.edu/wli/zipf/>.
- [16] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in *Proc. of the 1997 USENIX Symp. on Internet Technology and Systems (USITS'97)*, Dec. 1997, pp. 193–206.
- [17] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy, "Organization-based analysis of Web-object sharing and caching," in *Proc. of the 2nd USENIX Symp. on Internet Technologies and Systems (USITS'99)*, Oct. 1999.