Characterization of a Large Web Site Population with Implications for Content Delivery

Leeann Bent[†*], Michael Rabinovich[‡], Geoffrey M. Voelker[†], and Zhen Xiao[‡]

[†]University of California, San Diego
9500 Gillman Dr. MS-0114
La Jolla, CA 92093-0114 USA
{lbent,voelker}@cs.ucsd.edu

[‡]AT&T Labs-Research
180 Park Ave.
Florham Park, NJ 07932 USA
{misha,xiao}@research.att.com

ABSTRACT

This paper presents a systematic study of the properties of a large number of Web sites hosted by a major ISP. To our knowledge, ours is the first comprehensive study of a large server farm that contains thousands of commercial Web sites. We also perform a simulation analysis to estimate potential performance benefits of content delivery networks (CDNs) for these Web sites.

We make several interesting observations about the current usage of Web technologies and Web site performance characteristics. First, compared with previous client workload studies, the Web server farm workload contains a much higher degree of uncacheable responses and responses that require mandatory cache validations. A significant reason for this is that cookie use is prevalent among our population, especially among more popular sites. We found an indication of wide-spread indiscriminate usage of cookies, which unnecessarily impedes the use of many content delivery optimizations. We also found that most Web sites do not utilize the cache-control features of the HTTP 1.1 protocol, resulting in suboptimal performance. Moreover, the implicit expiration time in client caches for responses is constrained by the maximum values allowed in the Squid proxy. Finally, our simulation results indicate that most Web sites benefit from the use of a CDN. The amount of the benefit depends on site popularity and, somewhat surprisingly, a CDN may increase the peak to average request ratio at the origin server because the CDN can decrease the average request rate more than the peak request rate.

Categories and Subject Descriptors

C.2.5 [**Computer Communication Networks**]: Local and Wide Area Networks—*Internet*; C.4 [**Performance of Systems**]: Performance Attributes; I.6 [**Simulation and Modeling**]: Applications

General Terms

Measurement, Performance

Keywords

Web caching, Content distribution, HTTP, Workload characterization, Cookie

[*]Bent performed this work while at AT&T Labs-Research.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-844-X/04/0005.

1. INTRODUCTION

With the enormous growth of Web traffic on the Internet, various technologies have been proposed to optimize the amount of bandwidth consumption due to Web accesses, including caching, prefetching, and content delivery networks (CDNs). Understanding the characteristics of Web workloads is essential for evaluating the benefits of these various content delivery technologies. However, although there has been considerable effort characterizing Web client workloads at many scales and locales (e.g., [10, 11, 12, 15, 22, 27]), there has been little previous work characterizing Web site workloads across a large numbers of Web sites. Previous work in this area has either considered a small set (under a dozen) of hand-picked Web sites [4], or a single high-volume Web site [3, 13, 21, 23]. While these studies have been valuable for certain purposes, a comprehensive study to a broader, more representative population of Web sites can provide important insights for research in content delivery as well as Web site design.

In this paper, we present a systematic performance study of a large number of Web servers: We examine the traffic to three thousand commercial Web sites hosted on a large server farm by a major Internet service provider. Unlike previous proxy-based studies that focus on traffic analysis from a client/cache point of view, we present a server-eye view of the properties of these Web sites. We also perform a CDN simulation analysis to estimate potential performance benefits a Web site might see from subscribing to CDN services.

We make several interesting observations about the current usage of Web technologies and Web site performance characteristics. Compared with previous client workload studies, the Web server farm workload contains a much higher degree (66% of responses) of uncacheable responses and responses that require mandatory cache validations. A significant reason for this is that cookie use is prevalent in our workliod (47% of requests), especially among more popular sites. Further, we found an indication of wide-spread indiscriminate usage of cookies, which impedes many content delivery optimizations unnecessarily. We also found that most Web sites do not utilize the cache-control features of the HTTP 1.1 protocol, resulting in suboptimal performance. Moreover, the effective expiration time of most of their responses in client caches (the "time-to-live" or TTL) is constrained by the maximum values allowed in the popular Squid proxy. Finally, our simulation results indicate that the Web sites benefit from the use of a CDN, with the amount of the benefit depending on the site popularity. At the same time, somewhat surprisingly, the peak to average request ratio may be increased when using a CDN because the CDN can decrease the average request rate more than the peak request rate.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents our trace methodology. Section 4 describes general properties of our server population. Section 5 presents a per Web site analysis, and Section 6 studies the benefits of using a CDN for the Web servers. Finally, Section 7 concludes.

2. RELATED WORK

A summary of Web characterization studies can be found in [24]. Many studies examine the performance of a small set (on the order of a dozen) of Web sites in detail, including [4, 20, 7]. Other studies present a careful analysis of a single very high-volume site [3, 13, 23]. Unlike these studies, ours is a systematic study of a large number of commercial Web sites, focusing on the busiest 3000 sites of a Web site population totaling over 34 thousand sites. Furthermore, unlike the previous studies, we did not manually select Web sites to examine but included the sites hosted on a large server farm. Thus, our study provides a more representative view of the properties of "run-of-the-mill" commercial sites. Another difference is that some previous work (e.g., [23]) only considers accesses to root pages, while our analysis includes accesses to all objects.

Krishnamurthy and Arlitt [16] and Krishnamurthy and Wills [19] examine accesses to many Web sites. However, [16] focuses primarily on protocol compliance, while [19] focuses on persistent and parallel connection usage. In addition, both papers study a far smaller number of Web sites than our study and only consider root pages.

Many studies analyze accesses to a large number of Web sites from a client-centric point of view (e.g., [5, 28]). While client-centric studies provide insights into the behavior of a given set of clients, only a study from a server perspective, such as a large server farm, can fully capture the set of events that happen at the server.

Various studies report some of the characteristics that we study, albeit from a client perspective. Feldmann et al. [11] reported the frequency of cookie occurrence and cacheability of Web content. Wills and Mihkailov focused on content cacheability [26]. Using a set of over a thousand URLs selected from a proxy log, they observed that many requests to images carry cookies that are functionally unnecessary: the cookies do not affect responses. This observation has significant implications to the Web sites in our study.

The frequency of modifications to Web pages was considered by Douglis et al. [9] and Brewington and Cybenko [6]. We do not focus on this characteristic, but it is indirectly relevant to our study because it affects the prevalence of "Not-modified" responses.

Several studies consider CDN benefits. Krishnamurthy et al. concentrate on a comparative performance study of different CDNs [18]. Unlike our work, Krishnamurthy et al. are interested in the download time of pre-selected pages through various CDNs, whereas we consider the CDN effects on a Web site. Jung et al. investigate the ability of a CDN to protect a Web site from a flash crowd [14]. While they study known flash events that occurred on two Web sites, we consider a large set of Web sites during a random 20 hour time period.

Ranuak et al. studied the benefits of proxy caches on the tail of the load distribution [25]. They found that a proxy cache benefits the peak bandwidth intervals much less than average bandwidth intervals and conjecture this is due to poor locality at the peak. We find similar results for CDN usage considering CDN hit rate.

3. METHODOLOGY

Our trace consists of 21 hours of the first TCP data packet of HTTP requests and responses into and out of a Web server farm from a large commercial hosting service of a major ISP. The trace was gathered using the Gigascope appliance [8] from 11:00pm on July 14, 2003, to 7:49pm on July 15, 2003. It contains 41,943,804 requests and 38,828,393 responses comprising 47 GB of total data before processing. Because our trace interleaves separate request and response records, we have to match each request with the appropriate response. After matching requests and responses, we post-processed the trace for analysis.

3.1 Matching Requests and Responses

Our trace contains interleaved records for requests and responses. To examine the data as HTTP request/response transactions, we must match requests with responses. We perform this matching based on the five-tuple of source IP address, destination IP address, source port, destination port and time. We then sort request/response pairs by request time. After matching the requests with responses, the trace contained 26,136,345 request/response pairs.

To identify matching errors, we compared the type of the object request with the type of the object response. We used the URL suffix as a heuristic for the request object type, and the response type in the HTTP header for response object type. We have excluded control responses from this comparison, as many of these have no type (e.g., "304 Not Modified" is often given without object type) or a fixed type (e.g., "404 Not Found" returns HTML regardless of request type).

To validate our matching procedure, we tested one and a half hours from the trace. We found a response type for more than half of the responses (57%), and found a 1.4% mismatch rate after the post-processing step described below. Since this error rate is reasonably small, we considered our matching procedure accurate enough to perform our analyses.

3.2 Post-Processing

To gather commercial Web site statistics from the trace, we post-processed the request/response pairs. We excluded all but the top 3000 sites. We excluded unpopular sites because they had a request rate that was too low – fewer than 485 requests per site over the course of the trace – to draw meaningful conclusions.

We also removed requests/response pairs that could not be mapped to a single Web site. Since some IP addresses in our study serve multiple Web sites and some Web sites have multiple IP addresses, we cannot use IP address as an indication of Web site. Instead, we use the `Host` field given in the HTTP header when available. Below we discuss how we use this field in more detail.

Note that we define *hostname* to be the actual host listed in the `Host` HTTP header field, while we define *Web site* to be a set of hostnames sharing the last two components of their domain names. For example, `www.firm-x.com` and `images.firm-x.com` are hostnames associated with the Web site `firm-x.com`. Because we are using the `Host` field to identify Web sites, we removed all requests that did not identify a hostname in the `Host` field. There were 90,301 (0.3%) request/response pairs in the trace with no `Host` field. Further, pipelined request/response pairs were removed. We identified these by looking for request packets with multiple `GET` requests in them. While pipelined request/response pairs may contain a valid match (the first `GET` request may be a correct match with the response), the remaining `GET` requests will have no match. Since there were very few of these we have removed them, along with requests that contain multiple `Host` HTTP headers. The total number of request/response pairs that were removed due to pipelining or multiple host names is 1510 (less than 0.1% of request/response pairs).

After this, we identified *Web sites* by first cleaning up the `Host`

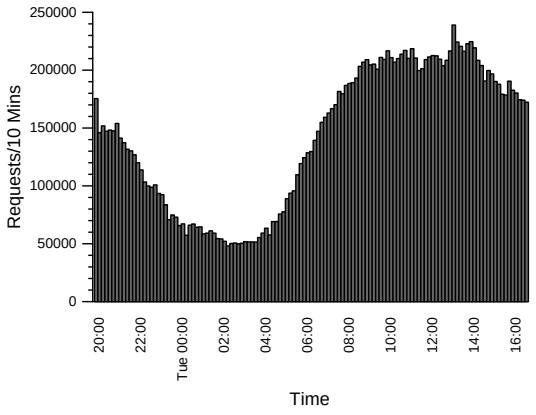


Figure 1: Requests to the 3000 most popular Web sites over time for the duration of our trace.

HTTP field (discarding those that could not be cleaned) and then by associating *hostnames* with *Web sites*. We found that some request/response pairs have given the `Host` HTTP header as an IP string. Where appropriate (i.e., there was a unique mapping of hostname to IP address), we mapped the IP address to a hostname. `Host` HTTP headers where there was no mapping were discarded. There were 153,355 (0.6% of request/response pairs) of these. We also found that there were request/response pairs with partial hostnames (because the packet had been cut off). If a hostname is a proper prefix of another hostname it is deemed a partial hostname. Requests with partial hostnames were also removed. There were 6,468 such request/response pairs (less than 0.1% of the total). Finally, hostnames with unprintable characters (there were only 27 of these) were removed.

Once the `Host` field had been cleaned, it was used to group the hostnames into Web sites using the last two components of the hostname as described earlier.

4. TRACE PROPERTIES

We start our analysis by summarizing overall characteristics of the workload in our trace. After post-processing, the trace contained 17,818,437 matched request/response pairs to 3,000 Web sites. Figure 1 shows these requests to the Web sites over time at the granularity of 10 minutes. This timeseries exhibits the diurnal pattern exhibited by typical Web workloads. The average request rate per Web site was 5939 requests over the lifetime of the trace, or 282 requests per hour. The Web sites in the trace span a wide range of activity, with the most popular site receiving 2.1 million requests and the least popular receiving only 485. In Section 6, we explore the topic of origin server load by studying the effects of using a CDN with this Web server farm.

Table 1 shows various overall properties of our trace. In terms of protocol version, we found that 14% of requests used HTTP/1.0 and 86% of requests used HTTP/1.1. The average response size was 9 KB for those objects that had a non-zero sized response.

Table 2 shows requested object types. Most requests (77%) were to images (URLs with file extensions of `.gif/.jpg/.jpeg`), 5.2% were to HTML (URLs with file extensions of `.html/.htm`) or base pages (i.e., `/`), 8.6% are CGI, and very few (0.2%) are documents (URLs with file extensions of `.ps/.doc/.ppt/.pdf`). We identified CGI objects as those objects whose URL contains the `/cgi` substring (used as an indication of the `/cgi-bin/` path or other cgi directory), contains a question mark (`?`), or ends in `.asp`, `.aspx`, or `.cgi`.

Property	Value
Request/Response Pairs	17,818,437
Trace size	33GB
Number of Clients	376,678
HTTP/1.0 downloads	14%
HTTP/1.1 downloads	86%
Average Object Size	9KB

Table 1: Overall trace properties.

Object type	Prevalence
Images	77%
HTML	5.2%
CGI	8.6%
.doc,.pdf,.ppt,.ps	0.2%
Audio	0.1%
Other	9.1%

Table 2: Request object type popularities.

Response type	Prevalence
Ok (200)	64%
Not Modified (304)	29%
Found (302)	2.6%
Client Error (4XX)	3.9%
Other	0.4%

Table 3: Response code popularities.

Compared to client workloads, this server farm workload contains a much higher concentration of requests to images (77% in our workload compared to 54% in [27]). We speculate that the nature of the commercial Web sites results in a larger percentage of image content compared with all Web sites accessed by a large client population. The preponderance of image requests has substantial caching implications for our workload, and discuss this issue in detail in Section 5.1 below. Also, the average response size in our trace is significantly lower than in previous studies (9 KB in our trace compared with, e.g., 15 KB in [5]). This effect partly can be explained by a large number of small embedded images.

Table 3 shows the relative prevalence of response codes in the trace. The majority (64%) are `200 OK`, and most of the remainder (29%) are `304 Not Modified`. We found only a negligible number of `5XX Server Error` responses. Note that `304` responses are caused by validation requests that ask for the content only if it has changed from the version cached by the requestor. The large percentage of `304` responses indicates that many of these validation requests were in a sense unnecessary because the requested object at the server has not been modified, and the cached object could have been used. We explore this issue in more detail in Section 5.2 below.

Finally, an interesting question is how often requests to our Web sites come through cache servers. According to HTTP/1.1., cache servers must include a `Via` header as they forward client requests upstream. While it is unclear how often this provision is adhered to, the `Via` header providers a lower bound on the number of requests coming through cache servers. About 8.5% of requests in our trace include the `Via` field. Additionally, we found that 2511 of 3000 (83%) Web sites see at least one request from a downstream cache.

Thus, while proxy caching or forwarding is not prevalent among our client population, most sites do indeed see some form of downstream caching, enough to justify the use of cache-controlling features of the HTTP protocol.

5. PER WEB SITE ANALYSIS

One of the main goals of our study was to characterize a large population of Web sites, particularly with respect to the properties relevant to content delivery. We focus on metrics that influence performance optimizations and may indicate potential performance problems within a web site. We do this by analyzing behavior on a per web site basis. Average behavior gives us intuition about the behavior of a commercial web site. Outliers from any average metric are good candidates for performance problems. Per web site statistics will show where we get the biggest win for potential performance improvements, on a per server basis and from a "network" point of view. Finally, the per web site analysis highlights unexpected behavior of web servers and unintended consequences of web server policies.

5.1 Cookie Usage

We start by studying the usage of cookies by the Web sites in our trace. A cookie is a piece of data that the server includes with its response to the client and which the client sends back to the server in subsequent requests. Cookies are widely used in today's Web to personalize content, to track user browsing within the site, to prevent unauthorized access, etc.. To store a cookie on a client, the server uses a `Set-Cookie` HTTP response header. Subsequent requests will carry the cookie in a `Cookie` HTTP request header. Besides the cookie value, the `Set-Cookie` header includes `domain` and `path` attributes, which determine the URLs that are relevant to this cookie. Only requests for URLs with host names belonging to the specified domain and with URL path prefixes matching the path attribute will include the corresponding cookie header.

Cookie usage has important implications for content delivery. Cookies are often used for page personalization or for e-commerce, and thus "cookied" requests (i.e., requests that carry a `Cookie` header) are usually not satisfied from caches, be it forward proxies or CDN edge servers. At best, caches send `If-Modified-Since` requests with the same cookie to origin servers before sending their cached responses to clients. This is true despite the fact that HTTP/1.1 formally allows using cached responses for cookied requests unless expressly prohibited by cache-controlling headers. Thus, a cookied request from a browser always reaches the origin server either as a full request or as an `If-Modified-Since` request. Because of their "cache busting" effect, cookies must be used judiciously to fully exploit Web caching infrastructure.

5.1.1 Frequency of Cookie Use

We find that cookie use in the requests in our trace is widespread: 47% of requests contain a cookie. Furthermore, cookie use is skewed toward popular web sites. While 47% of requests contain cookies overall, only 35% of servers receive requests with cookies in them. This implies that those 35% of servers receive a disproportionally high share of requests. To illustrate this trend in more detail, Figure 2 splits all sites into popularity bins, where the left-most bar corresponds to the most popular 1% of web sites, the next bar corresponds to the next most popular 1% of web sites, and so forth. The bar heights correspond to the percentage of cookied requests directed to the corresponding groups of servers. This graph clearly shows that cookie use is skewed toward the more popular sites, especially the most popular 1% of websites. We also performed a correlation analysis between the site popularity (measured in the number of requests to the site in the trace) and the prevalence of cookies. When considering the 3000 most popular sites, we found a low positive correlation of 0.09. Analyzing just the 279 sites with at least 5000 requests, we found the significant positive correlation of 0.18.

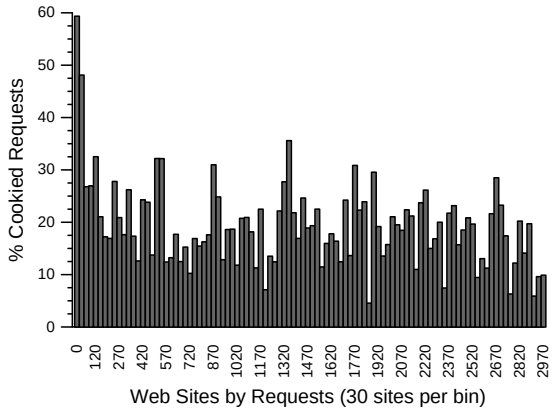


Figure 2: Prevalence of Cookie Use by Server Popularity.

In summary, 47% of our workload is not fully cachable simply because they have a cookie attached. Our next question is if this high usage is inherently necessary. Obviously, discerning the intention of Web site designers is imprecise science, and so answering this question authoritatively is difficult. However, we can evaluate the extent to which cookie use could be changed to improve the delivery of a site's content. We consider two closely related aspects as an indication of unnecessary usage.

First, we consider cookied requests to images as an indication of unnecessary use. Wills and Mikhailov observed that almost 90% of cookied requests for images return responses that do not depend on the cookies and concluded that cookies in most of these requests are not inherently necessary [26]. Indeed, if not for image personalization, cookies in image requests could also be used to track accesses by a given client. But this tracking is redundant for embedded images since their accesses can be inferred from the accesses of their containing page. Only when the image is hyperlinked to a page and must be clicked on to be viewed can cookies be justified. Intuitively, images are more often embedded than hyperlinked (although sites serving primarily image collections such as adult sites are a notable exception to this rule).

Second, we consider responses that set cookies for all requests to any URLs on the site as an indication of injudicious cookie use. We conjecture that these non-specific `Set-Cookie` headers are largely responsible for the prevalence of cookied image requests mentioned earlier. In these sites, only the initial requests from a given client do not carry cookies. It is hard to imagine that every request to a site, including all applets, Javascript modules, images, etc., requires a cookie.

5.1.2 Cookied Requests for Images

Figure 3 shows the prevalence of cookied requests for images. The graph includes the approximately 1000 sites that use cookies. The x-axis shows the Web sites ranked in the order of their percentage of cookied requests that are image requests out of the total number of cookied requests, and the y-axis shows the percentage of cookied requests to that site that access images. We see a wide range of prevalence of cookied image requests. However, clearly a large number of sites set cookies so that they apply to images. This

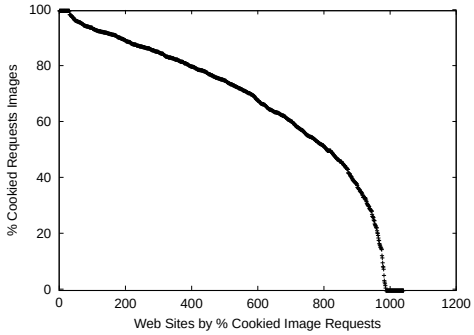


Figure 3: Prevalence of cookied requests for images.

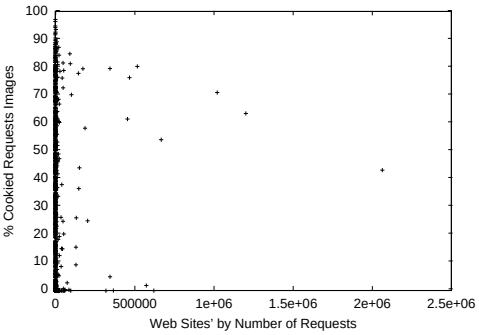


Figure 5: Prevalence of cookied requests for images.

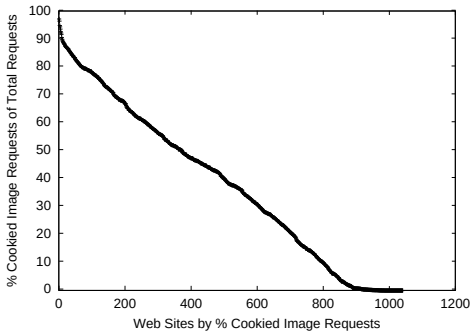


Figure 4: Percentage of cookied image requests over all requests to sites that use cookies.

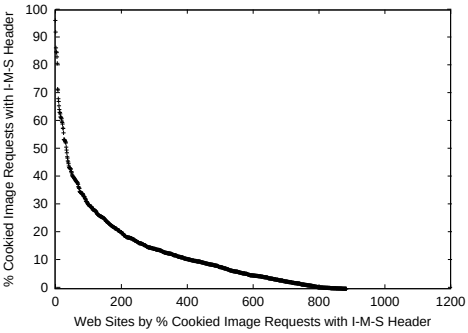


Figure 6: Percentage of `If-Modified-Since` cookied requests for images over all requests to sites that use cookies.

indicates a high occurrence of unnecessary cookie use: 73% of all cookied requests are for images and, overall, 34% of all requests are for cookied images. In other words, more judicious use of cookies would likely cut their concurrence by more than half, substantially improving the effectiveness of CDN and client-side caching.

In fact, we found that cookied image requests constitute a large portion of *all* requests received by those sites that use cookies. Figure 4 shows the percentage of all requests received by sites that were requests to images with cookies. The x-axis plots site rank in decreasing order of the above percentage for those sites that use cookies, and the y-axis shows the percentage value. We see that, for a large number of sites, cookied image requests represent the majority of the overall requests these sites receive. Given that images are responsible for a majority of requests on the Web in general (see, e.g., [2, 15, 28]), we can explain this result if we assume, again, that the main reason for the cookied images is that sites simply set cookies indiscriminately for all their URLs. But it also shows that, by doing so, these sites deny themselves much of the benefits from CDNs and Web caching.

The left-most point of Figure 4 is particularly interesting as it shows that practically all requests to this site were cookied images. A closer look revealed that this is a credit services site and most of the objects on the site are cookied images. Many of these images, in fact, appear to be navigational objects (e.g., menu bars) or spacer images in a menu or logo. Again, it seems unlikely that all of these images require cookies.

We also consider how indiscriminate cookie use depends on site popularity. While we showed that popular sites use cookies more often, we do not find that they use cookies more appropriately. Figure 5 shows how cookied image requests correspond to object popularity. On the x-axis, we show the number of requests received by each Web site and on the y-axis we show the percentage of cookied requests that are requests to images. As the outlying points of the graph show, even very popular sites (e.g., the sites with between 500K and 2.1M requests) return many cookied images.

As mentioned earlier, some clients mitigate the content delivery limitations of using cookies by issuing `If-Modified-Since` (IMS) requests for objects that they already have in their caches. Figure 6 shows that there are a large number of these requests. The y-axis shows the percentage of cookied image requests received that are IMS requests. The x-axis shows the Web sites that receive IMS requests for cookied images in decreasing order of percentage. Over 85% (884) of sites that use cookies receive at least one `If-Modified-Since` cookied image request. 14% of total requests are cookied IMS image requests. Recall that 34% of all requests to a site are cookied image requests. Then 20% of all requests were cookied image requests resulting in a full download. In addition, although the 14% requests with the IMS header do not consume much bandwidth, they still impact performance because they increase client latency and origin server load.

5.1.3 `Path` *and* `Domain` *Attributes*

The source of the injudicious cookie use can be traced to how sites specify `Set-Cookie` headers. Almost all `Set-Cookie` headers in the trace contain attributes that actually widen the applicability of a corresponding cookie from the path which set the cookie to all objects on the site.

For example, nearly all `Set-Cookie` headers contain the `path` attribute specifying the root path "`/`". With this attribute, any request to the host that set this cookie must carry the cookie. Notice that without a path attribute, the cookie would have only applied to requests for URLs that share the path with the URL that had set the cookie.

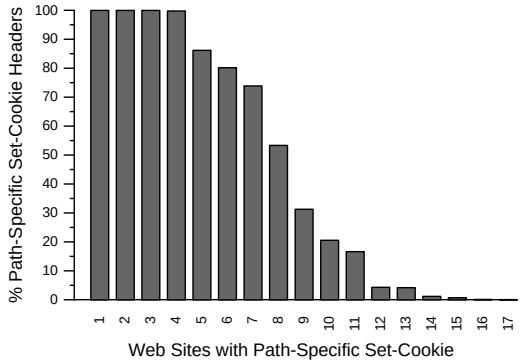


Figure 7: Prevalence of `Set-Cookie` header with restrictive path specifier.

Only 17 Web sites (out of around 750 Web sites we found using the `Set-Cookie` header) specify restrictive path attributes in `Set-Cookie` headers. Even these sites often provide these restrictive paths only in some responses and use root paths in other cases. Figure 7 shows, for each of these 17 sites, the percentage of `Set-Cookie` headers with restrictive paths. We see that only 3 sites consistently specify restrictive paths for their cookies, indicating that only they use cookies judiciously. A fourth site almost always specifies restrictive paths (99.8%), but even one non-restrictive path means that there is one cookie that applies to the whole site for at least one client.

Focusing on the `domain` attribute, only 16 sites specify the domain attribute in `Set-Cookie` headers. The vast majority rely on the default instead, meaning that the cookie will apply only to the host that set the cookie. For instance, if the cookie was set by `www.firm-x.com`, it will not apply to URLs with host name `images.firm-x.com`. One could in principle use this default rule to restrict cookie use by grouping all resources requiring cookies into distinct domain names from other resources. However, most Web sites (2752 sites, or 92%) in this trace use only one host name for the entire site, and providing or omitting the domain attribute does not affect cookie applicability.

Of these 16 sites, only 4 sites specify a domain that is a specific, 3-level domain. In fact, 12 of the 16 sites that do provide a domain attribute in their `Set-Cookie` headers appear to use it to widen the applicability of the cookie. Similar to the path attribute, these sites specify the most general domain they can (using only the two top-level domain names, such as `firm-x.com`), thus requiring requests to all subdomains to carry the cookie. Again, this is an indication of indiscriminate cookie use.

5.1.4 Discussion of Cookie Usage

Our results show that cookies are used indiscriminately in the sites that we studied, and that this usage significantly limits content delivery performance. How can sites fix this problem?

A direct approach for restricting the use of cookies to the content that needs it is to use the site namespace to separate cookied content. For example, since most images typically do not require cookies, placing them in a separate path (e.g., "/images") or on a different domain (e.g., "images.domain.com") easily separates them from content that may require cookies (e.g., HTML container pages that require secure access or tracking). With this approach, the site can easily specify the path and domain attributes in the `Set-Cookie` header to only use cookies for such content.

However, separating content using the namespace may be inconvenient to the developer of the content. In terms of organizing and maintaining content, for example, it may be most convenient to organize embedded images with the HTML pages that contain them. In this case, manually using `Set-Cookie` to pinpoint cookie use would likely be burdensome. Since many commercial sites create their content using Web site authoring tools, we see this as an opportunity for such tools to assist content developers in managing cookie use for their content.

Of course, some sites *do* require cookies for much of their content, including image content. For example, one of the sites in our trace is an adult content site whose requests were almost entirely cookied images. For our purposes, the important characteristics of such sites are that:

- They have a high occurrence of images that are hyperlinked rather than embedded in HTML pages. In other words, the browser does not download these images automatically – a user must click on a link to download an image.
- The site needs to keep track of per-client usage of the hyperlinked images to discern preferences of a given user.

Besides adult sites, museum sites and other sites with many hyperlinked images (e.g., NASA) may belong to this category. Still, we argue that even for these sites setting indiscriminate cookies for the entire site is not necessary. For example, in addition to the hyperlinked images, the NASA site contains many embedded images that do not need to be tracked because their usage can always be inferred from the usage of their containing HTML page. Again, tracking only the hyperlinked images can be achieved organizationally by placing them in a separate directory path and setting cookies for this path only.

5.2 Cacheability Properties

Cacheability of content plays a pivotal role in the effectiveness of the various traditional content delivery techniques that focus on storing static objects in the network for future use. These techniques include both client-side proxy caching and server-side CDNs and Web accelerators.

5.2.1 Caching Headers

HTTP/1.1 provides to the Web site developer expressive headers that allow fine-grained control of caching behavior. These headers are responsible to a large extent for the complexity of the protocol and of the cache and CDN servers. However, we found that neither client browsers nor Web sites make much use of these headers. Table 4 shows the percentage of requests that use any of the various cache-controlling headers available to them (note that the sum of all header and header directives do not add up to the 'Any' header line due to rounding errors). Table 5 shows the same for Web sites and responses. We find that, among requests, the `Cache-Control` values of `max-age` and `max-stale` are most prevalent, both accounting for 2.5% of requests. Of response cache controlling headers, we find that `no-cache`, specified either as a `Pragma` value or as a `Cache-Control` value is the most widely used, accounting for 4.7% of responses. However, all of the cache-controlling headers are used very little and only 6.8% of all requests and only 9.3% of all responses use *any* sort of cache control headers. In addition, the most prevalent response caching header is used to *deny* cacheability.

Table 6 shows the use of additional response headers affecting cacheability in the trace. The `ETag` header is used to validate cached objects. We see widespead usage of the `ETag` header, with

Header or Header Directive	Prevalence (overall)
Max-Age	2.5%
Max-Stale	2.5%
No-Cache	2.2%
Other	0.1%
Any	6.8%

Table 4: Usage of cache-controlling headers in requests.

Header or Header Directive	Sites Using the Header	Prevalence (among using sites)	Prevalence (overall)
No-Cache	175	9.1%	4.7%
Private	726	6.2%	2.1%
No-Store	30	4.4%	2.0%
Expires	215	10%	3.3%
Revalidate	32	9.2%	3.9%
Age	2	33%	0.2%
Max-Age	18	17%	2.0%
No-Transform	92	9.2%	0.1%
Other	27	3.0%	1.7%
Any	838	7.7%	9.3%

Table 5: Usage of cache-controlling headers in responses.

Header or Header Directive	Sites Using the Header	Prevalence (among using sites)	Prevalence (overall)
ETag	2989	89%	82%
Chunked	1306	7.1%	4.1%
Range Response	1949	0.7%	0.3%

Table 6: Usage of cache-affecting headers in responses.

almost all sites using the header and 82% of responses overall. We see little use of two other cacheability-affecting response headers: `Content-Encoding:Chunked` and `Content-Range`. We found that both response types were supplied by a large number of Web sites (1306 and 1949, respectively), but were almost unused overall, with chunked encoding accounting for 4.1% of responses and range responses accounting for 0.3% of responses.

More extensive use of cache-controlling headers could improve cacheability, decreasing load at the Web site. In the absence of these headers, caches tend to be very conservative in what they cache. Hence, these Web sites defeat to a large extent traditional content delivery technologies from which they might otherwise have benefited, including reduced bandwidth consumption and server load.

Sites may not use these headers either because content developers are not aware of their existence, are not aware of the potential benefits of using them, or find that the complexity of using them is too burdensome. As with cookie usage, this situation presents another opportunity for Web site authoring tools. Such tools already manage the complexity of the content itself, and could also help manage the complexity of using the cache control directives to maximize content cacheability and improve content delivery performance.

5.2.2 *Use of* `Last-Modified` *Header*

A header that is not strictly for cache control but that has a profound effect on caches is `Last-Modified`, which specifies the time of the last update to the object. In the absence of an explicit

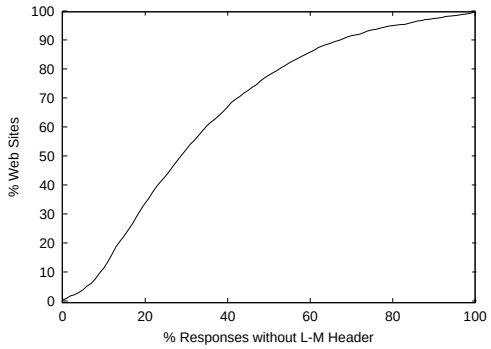


Figure 8: Prevalence of responses without the `Last-Modified` header.

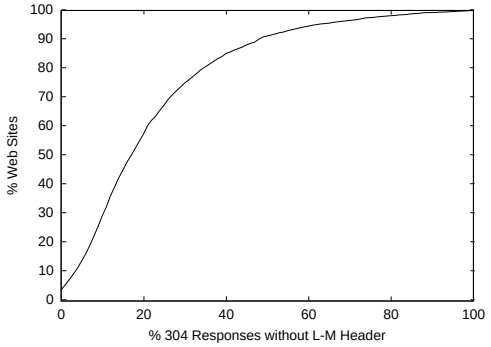


Figure 9: Prevalence of `304` responses.

expiry time, caches use the `Last-Modified` header to compute heuristically how long they can cache the response before validating it with the origin site. When there is no `Last-Modified` header, caches cannot use this heuristic. Generally, caches assume responses without the `Last-Modified` header were dynamically generated and conservatively do not cache them.

While almost all Web sites supply the `Last-Modified` header, the frequency of its use varies widely. Figure 8 shows the percentage of a site's responses without this header. Overall, a large fraction of responses (44%) lack this header. And, on average, 34% of responses from each site do not have the `Last-Modified` header. We also examined `Last-Modified` usage according to Web site popularity and found no correlation between Web site popularity and use of the `Last-Modified` header.

The large number of requests without `Last-Modified` may in itself indicate low content cacheability. However, a more detailed analysis shows that many responses with no `Last-Modified` header are "Not Modified" control messages that validate cache objects and do not affect content cacheability (see Figure 9). On average, 67% of responses without a `Last-Modified` header were "Not Modified" control messages (both on a per Web site basis and across all sites).

Because Web servers supply the `Last-Modified` header for static files automatically, we speculate that the lack of this header in a response (that is not a control message like "Not Modified") indicates a dynamically generated response.

Using the definition of CGI objects as in Section 4, Figure 10 plots the percentage of responses classified as dynamically generated according to this heuristic. Overall, only about a third of all sites (1195) return such responses, and, among these sites, these responses represent on average 4% of all responses from a given site

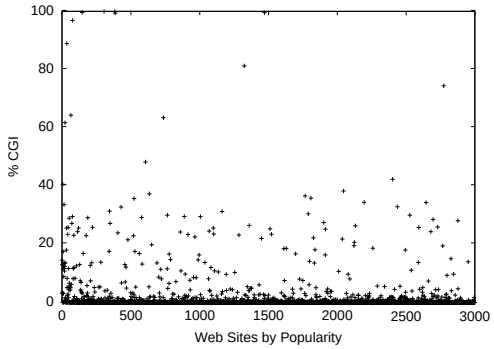


Figure 10: The percentage of requests to CGI objects for each Web site in decreasing order of site popularity.

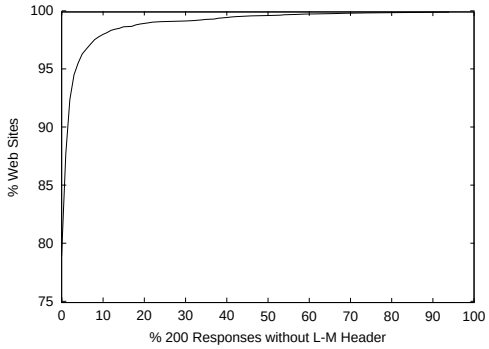


Figure 11: Prevalence of "200" responses other than CGI that have no `Last-Modified` header.

and 9% overall. Recall that 29% of responses were `304` responses with no `Last-Modified` header, leaving 38% of total requests without `Last-Modified` headers accounted for. However, this implies that 6% of requests overall still have no `Last-Modified` header. These requests are not `304` responses and are not classified as requests to dynamic content by the CGI heuristic above.

Thus the above heuristic undercounts some dynamic responses. To explore how this under counting affects individual sites, Figure 11 plots the distribution of `200` responses without the `Last-Modified` header that would not be identified as dynamic by the above heuristic (which we refer to as "non-cgi" responses for short). Over 98% of Web sites receive less than 10% of non-cgi `200` responses with no `Last-Modified` header. While this result seems to indicate low under counting, it is in fact quite significant compared to the number of requests identified as dynamic by the CGI heuristic. Moreover, a handful of sites (10) received more than 50% of non-cgi, `200` responses without the `Last-Modified` header.

5.2.3 Cacheability

Finally, we consider the overall frequency of responses that are not fully cacheable (i.e., responses that cannot be served from a cache without contacting the origin server). Following previous studies (e.g., [11]), we will call them "uncacheable" although, as discussed earlier, they may be only cache validations. Figure 12 plots the CDF of overall percentage of uncacheable responses. Note that a response can be uncacheable because of either the request or the response in the request/response pair. We found that a large portion of Web content from the sites in our trace is uncacheable: overall, 66% of responses across all the sites are uncacheable. Some-

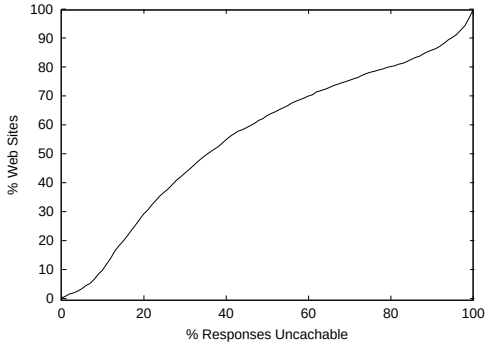


Figure 12: CDF of percentage of responses to a Web site that are uncacheable.

what surprisingly, this degree of uncacheability is substantially higher than that reported in previous client and proxy caching workload studies: Feldmann et al. [11] report 38–43% uncacheable responses from workloads of dialup modem clients at a large ISP and clients on a fast LAN, and Wolman et al. [27] report 40% uncacheable responses from a large university client workload. Per site, 45% of responses are uncacheable on average and 50% of Web sites have 36% or more responses uncacheable. We attribute this higher degree of uncacheability to our focus on commercial Web sites and substantial use of cookies.

Another conclusion is that CDNs have an opportunity to increase their benefits by a closer cooperation with content providers. For example, the content provider can explicitly invalidate cached objects in CDN caches, reducing the need for "Not Modified" responses without sacrificing data freshness. Also, by understanding the nature of the content, CDNs can be more aggressive in deciding what content can be cached. Finally, emerging technologies aimed at accelerating dynamic content and Web applications should further increase CDN benefits.

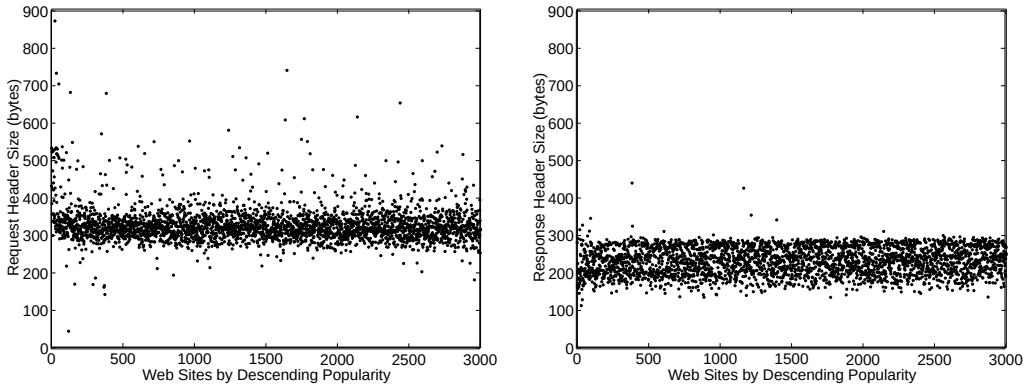
5.3 Cache Consistency-Related Properties

Figure 13 shows the cumulative distribution of the average TTL for the top 3000 Web sites. For all cacheable objects, we compute the TTL value for a response message based on the age and `Last-Modified Time` of the object using the same heuristics as in the Squid proxy [1], which enforces a maximum TTL of 4320 minutes (three days). The average TTL value across all responses is around 3730 minutes. Judging from the large number of the `304` requests shown in Figure 9, this limit might be excessive for many sites. Since reducing the limit has a danger of increasing the hit rate at web sites, it would be in content providers' interests to provide explicit expiration times for their responses since doing so would reduce the load on their servers due to 304 requests.

5.4 Response size, Header size, Compression

About 62% of response messages specify the `Content-Length` field. Overall, these response messages report an average content length of 8968 bytes. Figure 14 shows the scatter plot of average response size for the top 3000 Web sites in descending order of popularity (note that the y-axis is in log scale). We find little correlation between the popularity of a Web site and its average message sizes.

Figure 15 shows the header size distribution of request and response messages for the top 3000 Web sites in descending order of popularity. The majority (90%) of them have an average request header size between 270 bytes and 400 bytes, and an average re-



(a) Request Header (b) Response Header

Figure 15: Average header size for the top 3000 Web sites in descending order of popularity.

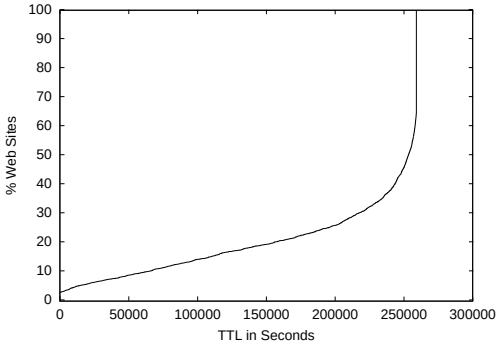


Figure 13: CDF of average TTL for the top 3000 Web sites.

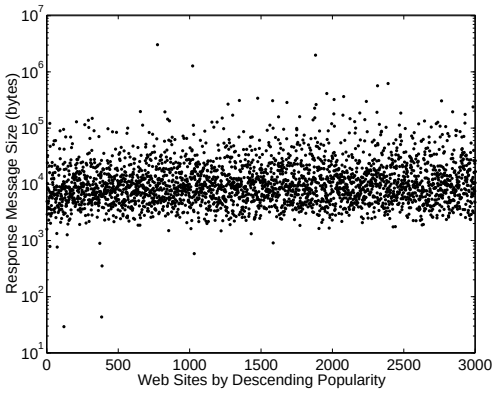


Figure 14: Average response size for the top 3000 Web sites. The y-axis is in log scale.

sponse header size between 170 and 282 bytes. If every Web site is weighted equally, the average header size is 325 bytes for request messages and 230 bytes for response messages. On the other hand, if we compute the average over the aggregate of all messages, the average size of request headers jumps to 402 bytes, while the average size of response headers remains relatively unchanged at 216 bytes. The largest average request header size we observed in the trace is 868 bytes, which is due to the use of cookies. Comparing Web sites according to their popularity, we see little correlation between the popularity of a Web site and the header sizes of its request and response messages.

The average request cookie size per Web site is 63 bytes, while across all requests the average Cookie size is 64 bytes. Most Web sites (90%) have an average cookie size between 12–130 bytes.

Overall, about 84% of requests indicate the willingness to accept a compressed response. This is consistent with the common impression that compression is widely supported in modern browsers. Those requests that do not support compression likely come from scripts or some out-of-date browsers. Somewhat surprisingly, only 8 out of the total 3000 Web sites ever return an object with a `Content-Encoding` header indicating some sort of compression. Even these sites returned on average only 5% of their objects compressed.

6. SIMULATION OF CDN BENEFITS

In this section we use trace-driven simulation to study the benefits of having the origin servers of the Web sites in our trace use a CDN to reduce their request load and bandwidth. We simulate the use of 20 CDN cache nodes based on a known CDN provider's configuration. Because of recent dramatic increases in disk sizes, we assume that CDN caches have unlimited cache capacity. We group clients into clusters based on network-aware clustering [17]. Since clusters group topologically close clients, we assume that CDN caches are assigned to clients at the granularity of clusters: all clients in the same cluster are assigned to the same cache. In our simulations, each client cluster is randomly assigned to one of the caches when it generates its first request, and uses that cache afterwards. In this study, our focus is on cache misses that reach the origin server, not latency effects as perceived by clients, and hence a different cache assignment will not change our results in any significant way.

In this paper, we study the upper bound of CDN benefits by assuming unlimited lifetimes for cached objects. This assumption make our results optimistic in terms of the request rate at the origin server because this request rate would only grow if objects were allowed to expire in the cache. Indeed, with infinite cached object lifetime and infinite capacity, each cache has a miss only once per object for the entire run.

First, we examine the impact of using a CDN on the peak request rate of the Web sites. One of the key benefits of using a CDN service is to reduce peak request load on origin servers. Figure 16

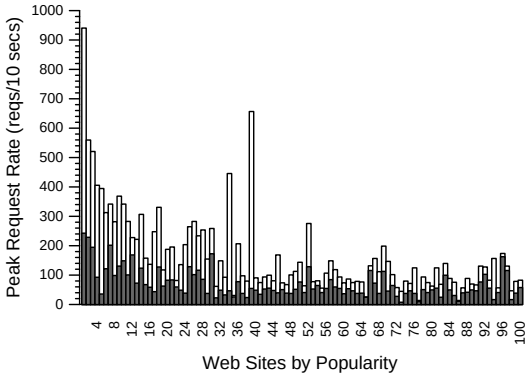


Figure 16: Comparison of peak request rate without a CDN (total height of each bar) and the peak request rate with a CDN (dark part of each bar).

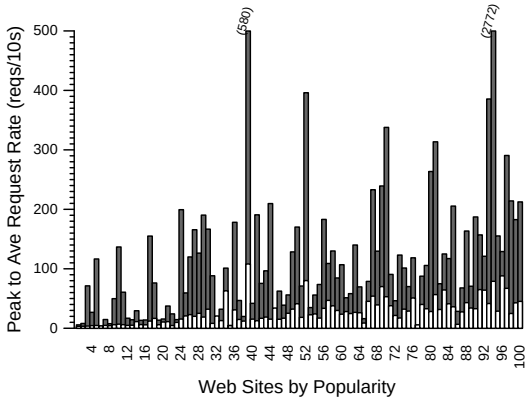


Figure 17: Comparison of peak to average ratio with and without a CDN. The white part of each bar shows the peak to average ratio for a Web site without a CDN, and the dark part shows the increase with a CDN.

shows the peak request rate of the 100 most popular Web sites with and without using a CDN. We compute the peak request rate for a Web site at the granularity of 10 seconds across the entire trace. In these analyses we focus on the top 100 Web sites because these sites are the ones most impacted by the use of a CDN. Each bar in the graph corresponds to a Web site, and the Web sites are shown in order of decreasing request popularity. The entire bar shows the peak request rate for that Web site across the entire trace without using a CDN. The dark part of each bar corresponds to the peak request rate when using the CDN. The white part of each bar shows the portion of the peak request rate that is handled by the CDN, and directly measures the benefit of using the CDN for that Web site in terms of the reduction in peak request rate.

From Figure 16 we see that a CDN can significantly reduce peak request rate for the origin servers for these Web sites. For example, the most popular Web site has its peak request rate reduced from 941 to 243 requests per 10 seconds, a reduction of a factor of 3.9. We also find that, as a general trend, the impact of the CDN decreases as Web site popularity decreases. For example, the CDN reduces peak request rate for the 10 most popular sites by a factor of 3.0 on average, a factor of 2.7 for the 30 most popular sites, and overall a factor of 2.4 for the 100 most popular sites.

Next, we study the benefit of using a CDN on the peak to average load on the origin servers for the Web sites in our trace. Peak to average load measures the resource over-provisioning required by an origin server to handle the burstiness of peak loads relative to the long-term average load. The higher the peak to average load, the more costly the over-provisioning required by an origin server. By using a CDN, a Web site can potentially reduce its peak to average load by having the CDN absorb the peak load. As a result, the origin server for the Web site requires fewer resources for provisioning.

Figure 17 compares the peak to average load for the 100 most popular Web sites with and without using a CDN. Again, each bar corresponds to a Web site and the bars are shown in decreasing order of popularity along the x-axis. In contrast to the previous graphs, though, the height of the entire bar corresponds to the ratio of peak request rate to average request rate *with* a CDN; the white part of each bar corresponds to the peak to average request rate at an origin server without a CDN, and the dark part corresponds to the additional peak to average request rate at an origin server when using a CDN. We compute the average request rate over the entire trace, and, as in Figure 16, the peak request rate over 10-second intervals.

We make two interesting observations from this graph. First, somewhat counterintuitively, a CDN increases the ratio of peak request rate to average request rate — dramatically, in some cases, by an order of magnitude. This, however, does not mean that a CDN makes things worse, but simply that a CDN decreases the average request rate even more than the peak request rate. One possible explanation for this result is that there is less sharing at the peak [25]. Another is that there is low pre-peak to peak sharing. In this scenario, a set of objects on the Web site that were previously unpopular suddenly become very popular. Thus, the observed peak is due to requests that populate the caches with the previously unpopular objects (as in [14]). Since many of these unpopular objects will be unseen prior to the peak, the CDN provides less benefit during this period. It is important to keep in mind that the absolute resource capacity required at the origin Web server is still smaller with a CDN. But using a CDN increases the relative burstiness experienced by the origin server, requiring higher relative over-provisioning.

Second, although there is variation, the general trend is a slight increase in peak to average request rate with a CDN for less popular Web sites (toward the right on the x-axis). For example, the peak to average request rate increases by 39 for the 10 most popular Web sites on average, by 54 for the 30 most popular Web sites, and overall by 109 for the 100 most popular Web sites.

In addition to reducing server load by alleviating peak request rates, CDNs also reduce the bandwidth requirements for Web sites. Reducing bandwidth is important economically for both individual Web sites and for the server farm itself. Typical pricing models charge sites according to bandwidth usage, so reducing bandwidth reduces cost for these sites. In turn, the server farm has to pay its ISP for bandwidth. If it pays for bandwidth consumed, reducing bandwidth directly reduces cost. If it pays for fixed bandwidth allocations, then reducing bandwidth minimizes the allocation it needs. Further, reducing bandwidth consumption reduces the computing and network resources required by sites to serve content.

To evaluate the impact of CDNs on bandwidth requirements, we study the extent to which our simulated CDN reduces the average byte rate of the Web sites in our trace. (Ideally, we would like to evaluate the effect of CDNs on peak byte rate also; however, since our trace does not include transmission times for responses,

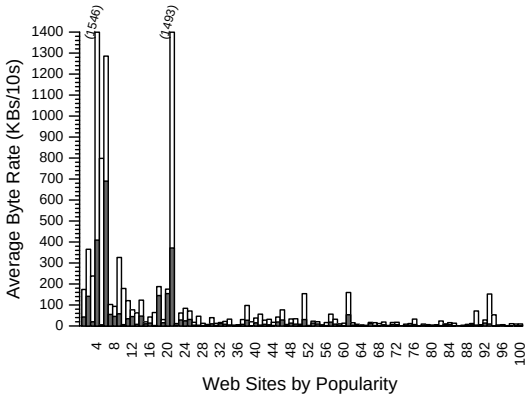


Figure 18: Comparison of average byte rate without a CDN (total height of each bar) and the average byte rate with a CDN (the dark part of each bar).

we could not compute this metric for this experiment.) Figure 18 shows the average byte rate for the 100 most popular Web sites in our trace with and without a CDN. We compute average byte rate for a Web site by first totaling the header and content lengths of all transactions to the site in the trace, and then dividing by the trace duration. Each bar corresponds to a Web site, and the height shows the average byte rate for the Web site without a CDN. The dark part shows the average byte rate when using a CDN; the white part shows the average byte rate handled by the CDN.

From Figure 18, we see that a CDN significantly improves average byte rate for these Web sites. Overall, using a CDN decreases average byte rate by a factor of 3.3 for the top 100 Web sites on average. For this metric, however, we do not see a trend related to the popularity of the Web site: the top 10 Web sites decrease average byte rate by a factor of 3.5, yet the top 30 Web sites decrease by a factor of 3.2.

7. CONCLUSION

Understanding the properties of Web sites is important for content delivery. This paper has investigated the characteristics of the 3000 busiest Web sites in a large server farm. We found that the Web server farm workload contains a much higher degree (66%) of uncacheable responses and responses that require mandatory cache validations. We found that many of the sites use cookies indiscriminately and fail to utilize the full cache-controlling features provided by the HTTP/1.1 standard. This results in suboptimal caching behavior for content delivery. We have also analyzed the benefit of content delivery networks using trace driven simulation. We found that CDNs can achieve significant reduction in bandwidth and request rate at the origin servers. Contrary to common belief, although the absolute request rate decreases, the peak to average request ratio may be increased when using a CDN because the CDN can decrease the average request rate more than peak request rate.

There are several directions for future work. We plan to extend our study to a larger time scale and analyze the dynamics of Web sites with respect to time. For example, it will be interesting to see how the traffic changes during different times of the day, during different days of the week, and the correlation between past and future traffic. We plan to add object cacheability information to our CDN simulation to estimate the benefits of CDNs under more realistic conditions. We also plan to enhance our simulation to estimate the benefits of so called overflow CDNs (i.e., CDNs used by a Web site only during periods of peak demand) and also model other content delivery technologies such as prefetching.

Finally, although numerous techniques have been developed, implemented, and standardized for improving the performance of Web content delivery, we find that most sites either do not take advantage of such techniques (e.g., cache control directives) or unknowingly inhibit them (e.g., indiscriminate cookie use). This situation arises because of the complexity of contemporary Web site content, the complexity of the techniques for improving content delivery, and the difficulty in measuring and evaluating the effectiveness of content delivery optimizations on Web sites. Without the ability to quantify the effect of their content delivery decisions, it is difficult for Web site maintainers to understand the implications of their decisions. One approach for addressing this situation is to develop a tool that gives Web site developers more insight into how their site performs and interacts with advanced content delivery mechanisms. We plan to investigate the design and use of such a tool as another direction of future work.

Acknowledgments

We are grateful to Oliver Spatscheck for stimulating discussions and for answering our numerous questions. We also thank the anonymous reviewers for their helpful comments. Support for this work was provided in part by AT&T support of the UCSD Center for Networked Systems. Voelker was supported in part by AFOSR MURI Contract F49620-02-1-0233.

8. REFERENCES

[1] The squid Web proxy cache. version 2.5. http://www.squid-cache.org.

[2] M. Arlitt, R. Friedrich, and T. Jin. Workload characterization of a Web proxy in a cable modem environment. Technical Report HPL-1999-48, Hewlett Packard Labs, Apr. 1999.

[3] M. Arlitt and T. Jin. Workload characterization of the 1998 World Cup Web site. Technical Report HPL-1999-35R1, HP Labs, Oct. 1999.

[4] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *Proc. of ACM SIGMETRICS*, pages 126–137, 1996.

[5] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web client access patterns: characteristics and caching implications. *World Wide Web*, 2:15–28, 1999.

[6] B. E. Brewington and G. Cybenko. How dynamic is the Web? In *Proc. of the 9th Int. World Wide Web Conference*, 2000.

[7] L. Cherkasova and M. Karlsson. Dynamics and evolution of Web sites: Analysis, metrics and design issues. Technical Report HPL-2001-1R1, Hewlett Packard Laboratories, July 16 2001.

[8] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: a stream database for network applications. In *Proc. of ACM SIGMOD*, June 2003.

[9] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: A live study of the World Wide Web. In *Proc. of the USENIX Symp. on Internet Technologies and Systems*, pages 147–158, Dec. 1997.

[10] B. Duska, D. Marwood, and M. J. Feeley. The measured access characteristics of World Wide Web client proxy caches. In *Proc. of the First USENIX Symp. on Internet Technologies and Systems*, pages 23–36, Dec. 1997.

[11] A. Feldmann, R. Cáceres, F. Douglis, G. Glass, and M. Rabinovich. Performance of Web proxy caching in heterogeneous bandwidth environments. In *Proc. of IEEE INFOCOM*, pages 107–116, 1999.

[12] S. D. Gribble and E. A. Brewer. System design issues for Internet middleware services: Deductions from a large client trace. In *Proc. of the First USENIX Symp. on Internet Technologies and Systems*, pages 207–218, Dec. 1997.

[13] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, 2(1-2):85–100, June 1999.

[14] Y. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In *Proc. of the 11th Int. World Wide Web Conference*, May 2002.

[15] T. Kelly. Thin-client Web access patterns: measurements from a cache-busting proxy. In *Proc. of the Int. Workshop on Web Content Caching and Distribution*, 2001.

[16] B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol compliance on the Web: A longitudinal study. In *Proc. of the 3rd USENIX Symp. on Internet Technologies and Systems*, pages 109–122, 2001.

[17] B. Krishnamurthy and J. Wang. On network-aware clustering of Web clients. In *Proc. of ACM SIGCOMM*, Aug. 2000.

[18] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proc. of the First ACM SIGCOMM Internet Measurement Workshop*, pages 169–182, Nov. 2001.

[19] B. Krishnamurthy and C. E. Wills. Analyzing factors that influence end-to-end Web performance. *Computer Networks*, 33(1–6):17–32, 2000.

[20] S. Manley and M. Seltzer. Web facts and fantasy. In *Proc. of the USENIX Symp. on Internet Technologies and Systems*, pages 125–133, Dec. 1997.

[21] J. C. Mogul. Network behavior of a busy Web server and its clients. Technical Report 95/5, Compaq Western Research Lab, Oct. 1995.

[22] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *Proc. of ACM SIGCOMM*, pages 181–194, 1997.

[23] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy Web site: Findings and implications. In *Proc. of ACM SIGCOMM*, Aug. 2000.

[24] J. E. Pitkow. Summary of WWW characterizations. *World Wide Web*, 2:3–13, June 1999.

[25] M. S. Raunak, P. J. Shenoy, P. Goyal, and K. Ramamritham. Implications of proxy caching for provisioning networks and servers. In *Proc. of ACM SIGMETRICS*, pages 66–77, 2000.

[26] C. E. Wills and M. Mikhailov. Examining the cacheability of user-requested Web resources. In *Proc. of the Fourth Int. Workshop on Web Content Caching and Distribution*, Apr. 1999.

[27] A. Wolman, G. M. 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 USENIX Symp. on Internet Technologies and Systems*, 1999.

[28] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative Web proxy caching. In *Proc. of ACM SOSP*, pages 16–31, Dec. 1999.