The Content and Access Dynamics of a Busy Web Site: Findings and Implications

Venkata N. Padmanabhan*
Microsoft Research
padmanab@microsoft.com

Lili Qiu†
Cornell University
lqiu@cs.cornell.edu

ABSTRACT

In this paper, we study the dynamics of the MSNBC news site, one of the busiest Web sites in the Internet today. Unlike many other efforts that have analyzed client accesses as seen by proxies, we focus on the server end. We analyze the dynamics of both the server content and client accesses made to the server. The former considers the content creation and modification process while the latter considers page popularity and locality in client accesses. Some of our key results are: (a) files tend to change little when they are modified, (b) a small set of files tends to get modified repeatedly, (c) file popularity follows a Zipf-like distribution with a parameter α that is much larger than reported in previous, proxy-based studies, and (d) there is significant temporal stability in file popularity but not much stability in the domains from which clients access the popular content. We discuss the implications of these findings for techniques such as Web caching (including cache consistency algorithms), and prefetching or server-based "push" of Web content.

1. INTRODUCTION

The exponential growth rate of World Wide Web has led to a dramatic increase in Internet traffic as well as a significant degradation in user-perceived latency while accessing Web pages. This has spawned significant research efforts aimed at reducing Internet bandwidth consumption due to Web accesses and improving user-perceived latency. Techniques to enhance Web performance, such as Web caching, prefetching, and server-based "push" of Web content, have been widely studied. Fundamental to the design and evaluation of these algorithms is a solid understanding of the characteristics of the Web workload and traffic.

There is a wide variety of commercial Web servers, including information servers (e.g., news sites), e-commerce servers (e.g., online bookstores), query server (e.g., search engines), upload server (e.g., sites where people submit their tax returns), and streaming server (e.g., Internet radio stations). Different types of Web servers may have very different workload characteristics, and hence may require different techniques for performance optimization. In this paper, we study the dynamics of an information server — the MSNBC news site [22] — which is one of the busiest Web sites in the Internet today. We also discuss broad implications of our findings for techniques such as Web caching (including cache consistency algorithms), and prefetching or server-based "push" of Web content.

The Web basically consists of servers, clients, proxies, and replicas. The servers are typically the originators of content while the clients[1] are the consumers of content. Proxies, when present, mediate the communication between a set of clients and a subset or all of the servers. Replicas are very much like proxies, except that they act on servers' behalf. For example, a content distribution network, such as Akamai, has multiple replicas to host heavily requested documents[2] for the Web servers. As such, each of these four components provides a unique perspective on the functioning of the Web. However, by far the majority of Web characterization studies have focused on data gathered at a proxy host (or by a network packet sniffer placed at a location where a proxy host might have been). Proxy-based studies are useful for many purposes: the design and evaluation of caching and prefetching algorithms, the characterization of server popularity, etc. However, proxy-based studies have their limitations because they offer only a limited perspective on the goings-on at clients and at servers, i.e., a proxy is not in a position to observe *all* of the events occurring either at clients (e.g., scrolling up and down in a browser window) or at servers (e.g., the servers' communication with clients that do not connect via the same proxy).

A significant difficulty that researchers face in doing a server-based study is the very limited availability of server traces. The few pioneering studies in this area [2] [3] have had to make do with data from relatively small departmental servers, typically at universities. While they have certainly been valuable, the main limitation of these studies is that the bulk of Web traffic is served out by large commercial servers. It is unclear how well inferences drawn from the

*http://www.research.microsoft.com/~padmanab/
†http://www.cs.cornell.edu/lqiu/

SIGCOMM '00, Stockholm, Sweden.

[1]We use the term "client" to refer both to programs such as Web browsers and to human users.
[2]We use the terms file, document, and object synonymously.

study of a small departmental server would scale to the large commercial sites in the real world.

We have been fortunate to have obtained access to detailed traces from MSNBC, a large commercial news site in the same category as the likes of CNN [8] and ABCNews [1], which is consistently ranked among the busiest sites in the Web [21]. The trace data we obtained was of two kinds: (a) *content logs*, which record file creation and modification events, and also include copies of successive versions of (a subset of the) files, and (b) *access logs*, which record client accesses to the HTML content (but not to the inline images). Aside from traces from several "normal" days, our data set includes traces from a couple of days that saw significant flash crowds because of a hot news story. The combination of content logs and access logs enabled us to study (certain aspects of) both the *back-end* (i.e., content dynamics) and the *front-end* (i.e., access dynamics) of the MSNBC site. Note that MSNBC did not make use of content distribution network (such as Akamai) during the period under study. Therefore the requests recorded in the accesses logs are sufficient to study access dynamics of the MSNBC site.

A detailed discussion of the results appears later in the paper, but here are some of our key findings: (a) file modification, although more frequent than file creation, often tends to change little in the file being modified, (b) a small subset of the files tends to get modified repeatedly, (c) file popularity follows a Zipf-like distribution with a parameter α that is much larger than reported in previous, proxy-based studies, and (d) there is significant temporal stability in file popularity but not much stability in the domains from which clients request the popular content. Our findings have implications for the effectiveness of Web caching and prefetching, cache consistency algorithms, and optimizations such as delta-encoding [20]. We discuss these in more detail later in the paper.

A limitation of our study is that it is difficult to determine how well the results derived from the MSNBC site generalize to the other large sites in the Internet. While there may certainly be characteristics that are peculiar to the MSNBC site, we believe that the operation of the site is quite typical of large news sites such as CNN and ABCNews. Therefore many of our findings and implications are likely to be applicable to other news sites. Our analysis of traces from the server end enabled us to study the dynamics of Web *content*, which is difficult to do with proxy traces. We believe that these points, coupled with the fact that it is challenging to obtain detailed *server-end* traces from large commercial sites (even from just a single such site!), make our study a valuable step towards characterizing the workload of such busy sites.

The rest of this paper[3] is organized as follows. In Section 2, we survey previous work. We present a discussion of the architecture of the MSNBC site, our trace collection methodology, and the traces themselves in Section 3. Then in Section 4 and 5, we present a detailed analysis of the content logs and the access logs, respectively. In Section 6, we summarize our key results and discuss the broader implications of our findings. Finally, in Section 7, we point out our ongoing and future work.

[3]This paper has been abbreviated to conform to the SIGCOMM page limit. A more complete version is available on-line [25].

2. PREVIOUS WORK

As discussed above, much of the work thus far in Web workload characterization has focused on proxies. In many instances, the goal is to evaluate the effectiveness of proxy caching. The hit rate of proxy caches have been found to be quite low, often not much higher than 50% [10] [13]. A substantial fraction of the misses arise from *first-time* accesses to files (i.e., compulsory misses). Proxy logs have also been used to study the effectiveness of cooperative caching. In this context, a recent study [29] [30] reports that the organizational membership of clients is significant in that clients belonging to the same organization are more likely to request the same documents than clients in different organizations. Our analysis of spatial locality (Section 5.3) shows this significance can be diminished, for instance, by the occurrence of a "hot" news event that is popular globally, i.e., across organizational boundaries.

The relative popularity of Web pages accessed via a proxy has also been studied extensively. The almost universal consensus is that page popularity follows a Zipf-like distribution where the popularity of the ith most popular file is proportional to $1/i^{\alpha}$. The value of α is typically less than 1, which makes popularity distribution as seen by the proxy rather flat (e.g., [5] reports that it takes 25-40% of pages to draw 70% of the client accesses). Our results (Section 5.4) show that while the Zipf-like distribution holds for the MSNBC server site as well, α tends to be much larger, typically 1.4-1.6. This rather steep distribution of file popularity suggests that replicating or *reverse caching* (which refers to placing a cache right in front of a server so as to intercept all incoming client requests) a small set of the server's files could cut down the server load significantly.

Proxy logs have also been used to study the rate of change and age distribution of files (e.g., [9]). Such information has been deduced indirectly using the last-modified timestamp in the HTTP response header, which opens up the possibility of missed updates. In contrast, we use file modification logs obtained directly from the MSNBC site back-end, so the possibility of missed updates is diminished or eliminated in our study.

Server-based studies are far fewer in number than proxy-based ones. A few of these have focused primarily on the network dynamics of busy Web servers [19] [4]. Others have focused on very specific aspects of the server's operation (e.g., [16] discusses the performance of reverse DNS lookups in the CNN server cluster [8]). These studies are interesting but orthogonal to the focus of this paper.

A few of the server-based studies have been along lines similar to this paper. [2] studied access logs from a set of Web servers, the busiest of which saw under 50000 accesses in a day. They showed that file popularity followed Zipf's distribution (i.e., Zipf-like with $\alpha = 1$). They also demonstrated the presence of temporal locality in file accesses. [3] studied various aspects of server behavior using data from a set of servers. They reported that 10% of the files accessed

accounted for 90% of the server requests, and that 10% of the (client) domains accounted for over 75% of the server requests. However, the busiest of the servers they studied only saw a *total* of around 350,000 accesses in a day. In contrast, the MSNBC server cluster sees, on average, over 25 million accesses each day to its HTML content alone (image accesses, which are not included in our traces, would increase this number significantly).

In summary, we see our study of the MSNBC site as complementing the existing body of literature on Web workload and traffic characterization. We believe both the extent of the dataset we have analyzed and the use of back-end logs to characterize the content dynamics make our study valuable.

3. EXPERIMENTAL SETUP AND METHODOLOGY

In this section, we briefly describe the essential aspects of the MSNBC server site, and discuss the trace data that we gathered and processed.

3.1 Server Site Architecture

The MSNBC server site comprises a cluster of over 40 server nodes, each running the Microsoft Internet Information Server (IIS) [18]. These server nodes are grouped together into sub-clusters containing approximately 6 nodes each. Load balancing is done at two levels. First, each sub-cluster is assigned a *virtual* IP (VIP) addresses and DNS round-robin cycles through the 6 VIP addresses for the entire MSNBC site. Second, within each sub-cluster that shares a VIP address, the Windows Load Balancing Service (WLBS) [18] is used to spread load evenly across the nodes. The main point to take away is that at different times, a particular client's request may be served by any of the 40 nodes.

3.2 Server Access Logs

Each server node maintains a standard HTTP access log that records several pieces of information for each client access: a timestamp, the client's IP address, the URL accessed, the response size, the server status code, etc. For administrative reasons, the server site operator chose to turn off logging for image accesses. So the logs only record accesses to HTML content. While the absence of image access logs is a limitation of our data set, we do not believe it interferes with our study in a significant way since we are still able to analyze access dynamics at the granularity of Web *pages* (i.e., the HTML content).

Table 1 summarizes the overall statistics of the server access logs we used in our study. For our analysis, we picked traces from several different periods, each spanning a few consecutive days. In some periods, we had only an hour's worth of traces per day, while in others we had traces from the entire day. The traces from 12/17/98 and 12/18/98 [4] were especially interesting, because these corresponded to a "hot" news event, namely the launching of *Operation Desert Fox* by the US military against Iraq. All the other traces in Table 1 are from 1999.

[4]Throughout this paper, dates appear in the format *month/day/year*.

The MSNBC server site saw, on average, over 25 million client accesses each day to its HTML content alone (image hits were in addition to this). We used all 40 server nodes when analyzing the 1-hour or the 3-hour long traces. However, due to disk and memory limitations, we only used logs from 9 or 12 (i.e., 22.5–30%) of the server nodes out of the cluster of 40 when analyzing the logs from a whole day period. Since requests are randomly dispatched to the server nodes, considering only a subset of the server nodes is unlikely to significantly bias or otherwise impact the results of our analysis. Moreover, most our results based on the partial set of server nodes are consistent with those based on all the server nodes.

In some of our analyses, we clustered clients together into *domains*, which we defined to be all but the hostname part of the clients' DNS names (e.g., the domain for foo.bar.com is bar.com). We determined the DNS name of a host via a reverse DNS lookup on its IP address. We had a fairly high success rate - typically over 70% - as reported in Table 1. For the analyses that involved domain information, we ignored clients for which the reverse DNS lookup failed. We realize that our definition of a domain is simplistic, and are currently looking at more sophisticated alternatives that also consider network topology.

3.3 Content Creation and Modification Logs

The back-end of the MSNBC site uses the Microsoft Content Replication System (CRS) [18] to replicate content from a staging server to each of the 40 server nodes. Each replication event is logged, specifying the time of replication and the file being replicated together with its size. However, all that a CRS log entry says is that a file was replicated, which by itself does not enable us to determine whether a new file was created or an existing one was updated. We disambiguate between file creation and modification by using several days' worth of CRS logs to prime our list of files that already exist, and thereafter (once the spike in the number of file "creation" events has subsided) treating CRS replication events for files not seen before as file creations[5]. The CRS system did not log file deletions, although, in general, it could have. The CRS logs we analyzed corresponded to the 4-week period from 10/1/99 through 10/28/99.

3.4 Content Logs

For a small fraction of the content hosted by MSNBC, we were able to obtain a log of the HTML content itself, i.e., successive versions of the files as and when they were modified. A new version of the file was logged on the hour if the file had been modified (at least once) in the past hour. The subset of files logged in this manner was determined by the server site operator[6]. The content log, although limited in scope, enables us to gain some insight into the evolution of files as they undergo modification.

[5]As a validation of this heuristic, we confirmed that the number of file creation events beyond the priming period does not diminish with time, as would have happened had the file creation events been "bogus".

[6]One of the reasons the site operator generated the content log was to feed it into various search engines for re-indexing the corresponding pages.

	12/17/98 - 12/18/98	8/1 - 8/5	8/3 - 8/5	9/27 - 10/1	10/7 - 10/11	10/14 - 10/18
Period	9 AM - 12 AM	10 - 11 AM	9 AM - 12 AM	all day	all day	all day
% Total server logs used	100	100	100	30	22.50	22.50
# HTTP requests	10413866	7061572	14183820	38191388	28102751	30000981
# Objects	34443	30199	35359	57053	60323	52429
# Clients	253484	440151	656449	1948609	1831027	1938437
% success for reverse DNS	58.587	-	76.227	78.967	78.344	-
# Domains	41025	-	75569	117615	396528	-
% GET requests	99.818	99.050	99.065	99.065	99.008	99.059
% POST requests	0.088	0.448	0.464	0.474	0.512	0.475
% HEAD requests	0.082	0.406	0.392	0.327	0.350	0.336
% Other requests	0.012	0.096	0.079	0.134	0.130	0.130
% OK responses	58.084	55.913	57.104	56.195	55.088	54.744
% Moved temporarily	4.554	15.017	15.267	17.647	16.047	18.443
% Not modified responses	36.946	27.529	26.231	23.812	26.517	24.501
% Unsuccessful responses	0.416	1.541	1.398	2.346	2.348	2.312

Table 1: Overall trace statistics

3.5 Proxy Logs

In some of our analyses (Section 5.4), we compare the access characteristics of the MSNBC server with that of a busy caching proxy. We obtained logs from a proxy cache that serves the Microsoft corporate campus with a population of over 50000 client hosts. The logs were gathered over a 2-day period - 10/6/99 and 10/7/99.

4. SERVER CONTENT DYNAMICS

In this section, we analyze the dynamics of file creation and modification. The content dynamics is important to study because it has profound implications for the effectiveness of Web caching, in particular, cache consistency control mechanism.

4.1 File Creation and Modification Processes

We studied the dynamics of file creation and modification using information derived from the CRS logs. We consider the time of replication of a file by CRS to be its creation/modification time because it is only after (the new version of) a file is replicated onto the server nodes that it becomes available to clients. There is a diurnal cycle as well as a weekly cycle to the file creation and modification processes. The diurnal cycle exists despite the global accessibility of the MSNBC Web site because the bulk of the accesses to the site come from North America, i.e., from time zones that are clustered together.

In Table 2, we tabulate a more detailed breakdown of the event counts during the one-week period from 00:00 hours on 10/9/99 to 23:59 hours on 10/15/99. We note there are nearly four times as many file modification events as creation events during the course of the week. A closer examination of the modification events reveals that they tend to be concentrated on a small number of files. On average, there were around 10 modification events per file (only considering files that were modified at least once during the week).

The large numbers of creation and modification events have broad implications. The creation of new files poses a challenge to latency/bandwidth saving schemes such as prefetching [24, 11] or server-initiated "push" [26]. These schemes depend on the past history of accesses to the file. But for a newly created file, there exists no such history. The large number of modification events suggests that it may be worthwhile deploying efficient protocol mechanisms for the validation/invalidation of files cached at proxies and/or clients (e.g., [6, 31]).

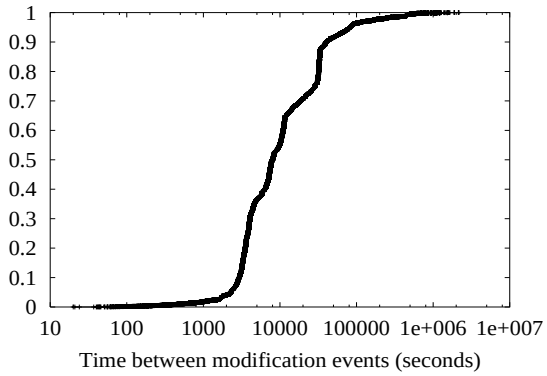
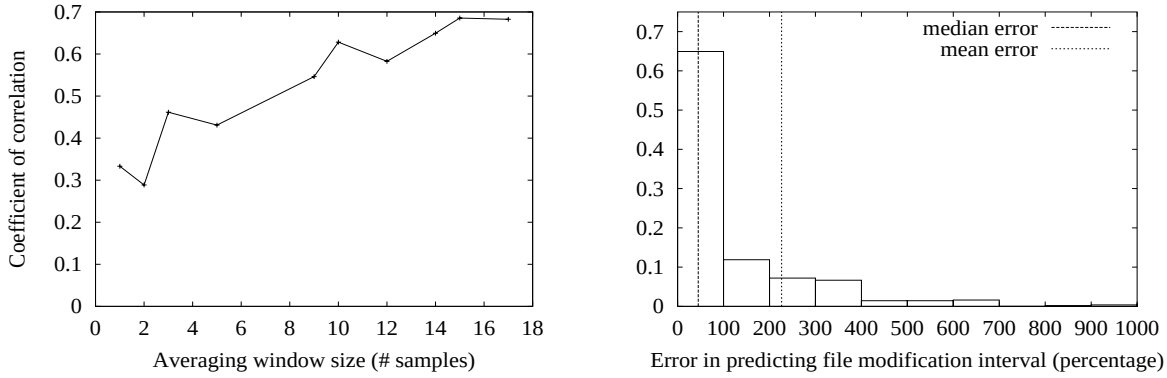


Figure 1: CDF of the time interval between successive modifications of a file (conditioned on the file being modified).

Creation	Modification	Unique Files Modified	GIF/JPEG Modification
6007	23343	2453	287

Table 2: Count of various events during the week from 10/9/99 through 10/15/99.

Table 2 also reveals that around 1% of the modification events corresponded to GIF/JPEG image files. Intuitively, we would not expect images to get modified very much; instead we would expect new images to be assigned new names. On closer examination, we discovered that the images being modified were almost exclusively maps, primarily weather maps but also maps of other kinds (e.g., a weekly "health" map indicating the current status of disease outbreak in the country). Interestingly, however, as found in [29], the cachability of images is lower than that of HTML due to server policy. For example, advertisements mostly consist of images, and are often not cachable.



(a) (b)

Figure 2: (a) The coefficient of correlation between the modification interval for a file and that predicted based on the average of past samples computed over various averaging window sizes. (b) A histogram of the percentage error in the prediction when the averaging window size is 16 samples. The median error is 45% and the mean error is 226%.

4.2 Distribution of Modification Intervals

Next, we turn to Figure 1, which shows the cumulative distribution function (CDF) of the time duration between two successive modifications (i.e., the *modification interval*) of a file. (Note that we only considered files that were modified during the 4-week period of our CRS logs. In other words, the CDF is conditioned on the file being modified during the 4-week period.) The CDF exhibits two distinct knees. The first is around the 5% level and occurs at a modification interval of around 3000 seconds (about 1 hour). The second is around the 95% level and occurs at a modification interval of around 80000 seconds (approximately 1 day). Both of these observations are in agreement with our intuition. By default, the CRS replication process is scheduled to run approximately once an hour. More frequent updates happen only when there are "hot" news events, which is typically an infrequent occurrence. Hence there are only a small number of instances of modification happening within an hour of the previous one. At the other end of the spectrum, a day seems like a natural "upper bound" for the update period. Of course, it is certainly possible for files to be modified on longer timescales such as weekly, monthly, etc. (or even aperiodically), but this is likely to be an infrequent occurrence.

4.3 Implications of Modification History

We now turn to examining the relationship between successive modification intervals of a file. We are interested in determining whether the modification dynamics of a file in the past is a good predictor of the future. This finding would have significant implications for Web cache consistency mechanisms such as adaptive TTL [14].

We estimate the new modification interval for a file to be the mean of past w samples of the modification interval where w is the size of the window used for averaging. Figure 2(a) shows the coefficient of correlation between the estimate and the true modification interval as a function of the averaging window size. We observed that the coefficient of correlation starts off low (around 0.25) when few samples are used to derive the estimate. As more samples are used, the estimate gets better. However, beyond 15 samples the coefficient of correlation tapers off at just under 0.7. Thus the modification interval for a file correlates moderately well with an estimate based on a sufficient number of past samples. However, inclusion of more samples does not help beyond a point.

In Figure 2(b) we take a closer look at the accuracy of an estimate of the modification interval of a file based on past samples. The figure plots a histogram of the error in the estimate (in percentage terms). The error lies within 100% (i.e., a factor of 2) and 400% (i.e., a factor of 5) in 65% and 90%, respectively, of all instances. The median error is 45%. This suggests that while accurate prediction of future file modification behavior may not be feasible, rough estimation is possible. For instance, if the average modification interval for a file based on past samples is 10 minutes, then we can be quite confident that the next modification of the file will happen say within an hour. While this is not a tight estimate, it would still be useful to know that a file is likely to be modified at the timescale of hours rather than days.

Our results suggest that past file modification behavior, when averaged over a sufficient number of samples, can yield a useful TTL setting (e.g., one hour versus one day) in the context of an adaptive TTL-based cache consistency scheme (e.g., [14]). However, the limited accuracy of the estimate means that there is the need for an alternative consistency mechanism (e.g., callback-based invalidation) as backup.

4.4 Extent of Change upon File Modification

We now turn to the question of how much a file changes upon modification. To answer this question, we use the content logs which, as described in Section 3.4, were available for a small subset of the HTML content. For two successive versions, v_1 and v_2, of a file, we compute the delta (i.e., difference) between them, $vdelta(v_1, v_2)$, using the *vdelta* algorithm [15]. Previous studies (e.g., [20]) have shown that the *vdelta* produces a more compact delta than most other algorithms. We quantified the extent of change between the

two versions of the file as $\frac{|vdelta(v_1,v_2)|}{\frac{|v_1|+|v_2|}{2}}$ where $|f|$ denotes the size of f. In other words, we quantified the size of the *vdelta* as a fraction of the average size of the pre-modification and post-modification versions of the file. The cumulative distribution function (CDF) of this quantity, expressed as a percentage, is shown in Figure 3. We observe that the extent of change tends to be small. In 77% of instances, the size of the *vdelta* is less than 1% that of the file undergoing modification. It is less than 10% of the file size in 96% of instances.

On closer examination, we found (at least) a couple of common modes of (minor) textual change that may explain these results: (a) a date/time string contained within the TITLE HTML tag is updated from time to time, and (b) the links to related pages are updated while leaving the bulk of the page unchanged. Moreover, when major modifications happen (e.g., when a new and fairly different version of a news story is generated), a new file (with a new name) is created rather than an existing file being modified.[7]

Our results suggest that techniques such as delta-encoding [20] would be very useful, especially in the context of slow links (e.g., dialup links).

4.5 Summary of Content Dynamics

In summary, our study of server content dynamics reveals that (i) files are modified and created frequently; (ii) the amount of modification is relatively small; and (iii) past modification behavior, with sufficient averaging, is a rough predictor of future modification behavior. These server dynamics have significant implications for the design of efficient protocols for the Web. In particular, it suggests that the potential benefit of an efficient cache consistency mechanism is large and that an adaptive TTL based scheme is likely to be useful.

5. SERVER ACCESS DYNAMICS

In order to further explore the impact of the server content dynamics on the Web traffic and to better understand the effectiveness of Web caching and prefetching, in this section we study the access dynamics seen by the MSNBC server site. Access dynamics is crucial to the effectiveness of Web caching. In particular, insights into the following issues may have a significant bearing on how we should improve the efficiency of the future Web:

- Relationship between server content dynamics and access dynamics, and potential benefit of cache consistency mechanisms
- Stability of user access patterns
- Significance of a user's domain membership
- The applicability of Zipf's law to Web requests at a server compared to the previous studies of requests seen at proxies

As shown in Table 1, over 99% of the requests employ the GET method. Moreover, among all the requests, the ones with HTTP response "OK" (status code 200) account for 55%. Around 15% to 18.5% of the requests have response "moved temporarily" (status code 302), and around 23% to 37% have response "not modified" (status code 304). The latter implies that the potential benefit of an efficient Web objects consistency protocol could be large. Unless otherwise specified, in our analysis of the access dynamics, we do not distinguish among different HTTP methods (i.e., GET, HEAD, and POST) and different status code. Instead we treat them all equally as Web accesses.

[7]Anecdotal evidence suggests that other new sites such as CNN do the same.

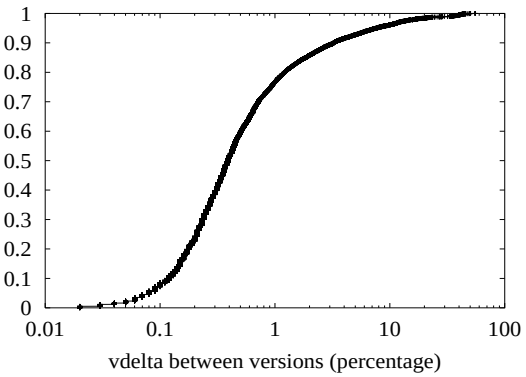


Figure 3: The CDF of the extent of change between succession versions of files. The extent of change is quantified as the ratio between the size of the *vdelta* difference between two versions of a file and the (average) size of the two versions. The ratio is expressed as a percentage.

5.1 Relationship between Server Content Dynamics and Access Dynamics

In this section, we analyze the relation between server content dynamics and access dynamics.

5.1.1 Correlation between age and popularity

First, we examined the correlation between document age (i.e., the time elapsed since its creation) and popularity. Our results are shown in Figure 4, where the x-axis denotes the time elapsed since document creation and the y-axis denotes the document ID sorted in decreasing order of popularity (i.e., document #0 is the most popular one). A dot at (x, y) in the figure means a document y is accessed at time x.

It is evident from the graphs that most documents receive a lot more accesses soon after their creation than afterwards. This happens because most news page are accessed when they are fresh. On the other hand, there are a number of documents (the ones denoted with a small document ID) that remain hot for the entire five-day period under study. These are typically index pages, such as the default front page for the MSNBC site.

5.1.2 Classification of accesses based on file creation and modification events

Previous research [27] has shown that up to 40% of accesses are to the objects that have not been accessed before by clients in the same domain. In a caching context, these lead

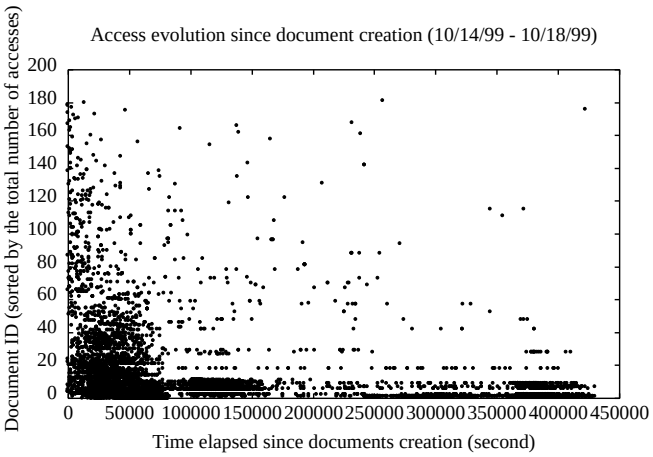


Figure 4: Distribution of accesses to documents since their time of creation.

to *first-time* (i.e., compulsory) misses. It is very useful to understand what comprises these first-time accesses/misses. Using both access logs and modification logs, we have found that most first-time accesses (which we determined on a per-domain basis) are to old objects that were created at least a day ago. This is evident from Table 3 (columns 2 and 3). We believe such accesses are very hard to predict. This is because objects that have not been accessed by a certain domain for several days following their creation are considered unpopular. Accesses to unpopular documents are hard to predict. Moreover because these documents are unpopular, it is not cost-effective to prefetch them in advance.

We also classify repeated accesses (i.e., accesses to previously requested documents by the domains) according to their modification history. Specifically, we divide the accesses into two groups: accesses to objects that are modified on the day of access, and those that are not. The former requires consistency control mechanism to ensure users always get the latest versions, while caching alone is sufficient for the latter type of accesses. As shown in Table 3 (columns 4 and 5), over half of the repeated accesses are to the modified objects. However the total number of modifications is several orders of magnitude smaller than the total number of accesses. This implies server-based invalidation can be a very efficient means of ensuring strong consistency, especially compared to client polling on every read [17, 31].

5.2 Temporal Stability of User Access

In this section, we analyze the temporal stability of Web accesses with the following questions in mind:

- How much does the popularity of Web pages change with time? That is, do popular Web pages on one day remain popular on the subsequent days?
- For each Web page, how much does the set of domains interested in it vary from one day to the next?

Answers to these questions are critical for designing sensible prefetching or server-initiated push algorithms. Any prefetching algorithm based on past history relies on a certain degree of stability, both in ranking and in the interest group (i.e., the set of domains requesting the pages). Our trace analysis helps shed light on how well such reactive prefetching algorithms might perform in practice.

5.2.1 Stability of Web Page Popularity Ranking

We study the stability of Web page ranking as follows. We consider Web access logs from several consecutive days. For each day, we pick the n most popular documents. We then compute the overlap in the most popular pages selected in one day to those selected in the next day. Our results are illustrated in Figure 5. On the x-axis is the number of most popular documents picked (e.g. $x = 10$ means we pick the 10 most popular documents), and on the y-axis is the percentage of overlap. Figure 5(a) plots the ranking stability in a week period. We make the following observations. First, the overlap is mostly over 60%, which essentially means many documents are hot on both days. Second, for the several consecutive days period, the overlap is mostly the same. For example, the amount of overlap between 8/1/99 vs. 8/2/99 is quite close to that between 8/1/99 vs. 8/5/99. This implies that the Web accesses within a short time-frame (such as a week) are equally good for predicting which Web pages will be among the most popular in the near future. That is, last week's trace is almost as useful as yesterday's trace for predicting Web pages ranking. This happens because many of the very popular pages are *index* pages (such as the default front page for the MSNBC site) that contain pointers to other pages. Third, the ranking stability tends to decrease as the number of documents selected increases. This indicates that very "hot" documents are more likely to remain hot than moderately hot documents. We observe similar results in the traces from other periods as well.

We also study the ranking stability in a larger time window considering the overlap between the two days that are more widely separated. Our results are shown in Figure 5(b). As we would expect, the overlap decreases as the time interval gets longer. For example, the overlap between 12/17/98 and 10/18/99, which are 10 months apart, is considerably smaller, mostly below 20%. On the other hand, even for the two months separation (8/1/99 and 10/18/99), although the overlap is lower than with a one-day separation, it is still quite significant. For the top 100 documents, the overlap is above 60%. However compared to the one day separation, the overlap in the two month separation decreases much faster as the number of documents selected increases.

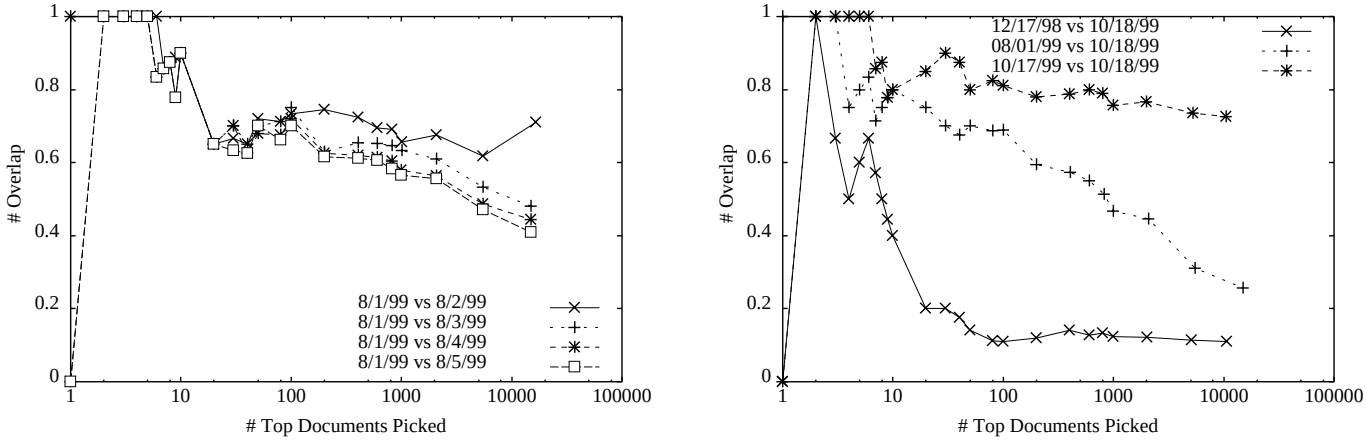
We further explore this issue by breaking down the overlap and disjoint regions into four groups (assuming Day 1 is prior to Day 2):

- *Common & unmodified*: documents that are popular on both days and have not been modified since Day 1
- *Common & modified*: documents that are popular on both days, and have been modified since Day 1
- *Different & old*: documents that are popular on only one of the days, but exist on both days
- *Different & new*: documents that are created after Day 1, and are popular only on Day 2

Our results are shown in Figure 6. We observe that many of the popular files in common between the two days are

Date	First & New	First & Old	Repeated & Mod		Repeated & Unmod
			# Mods	# Accesses	
10/8/99	72362	240119	7937	1317246	1014821
10/9/99	14652	96167	6051	697338	576113
10/10/99	15156	99248	6107	758626	610435
10/11/99	47619	206743	7324	1163424	919085

Table 3: Breakdown the Web accesses according to file modification/creation history.



(a) (b)

Figure 5: Stability of document popularity in both short and long period.

unmodified throughout the period (i.e., starting from day 1 through day 2). This implies that caching is potentially able to reduce Web traffic significantly. However, there exists a significant discrepancy between the number of files that are potentially cachable (i.e., not modified) and the number of files that are actually allowed to be cached by servers [29]. Bridging this gap will help reduce Internet traffic substantially as well as improve client latency.

Moreover the disjoint region in the set of popular files on the two days is mostly not due to the creation of new files, since most of the disjoint region consists of the accesses to files that are in existence on both days.

To summarize, in this section we studied the ranking stability of Web pages, and found that the stability is reasonably high on the scale of days. The ranking tends to change only gradually over time. We also examined what contributes to the overlap and disjoint regions of popular documents between a pair of days. Our results show the disjoint region mostly consists of old pages, not newly created pages. In the overlap region, the number of modified documents exceeds the number of unmodified documents, though both counts are of the same order of magnitude.

5.2.2 *Stability of Interest Group*

We now consider how the interest group for each Web page changes over time. Our approach is as follows: For each Web page that receives over 100 accesses on the first of the pair of days in our study, we find the set of domains which accesses the page on each day, and compute the overlap between these domain sets. As mentioned earlier, we ignore requests from clients for which the reverse DNS lookup fails. Since the percentage of failure is reasonably low (Table 1), this should not make a significant difference in our results.

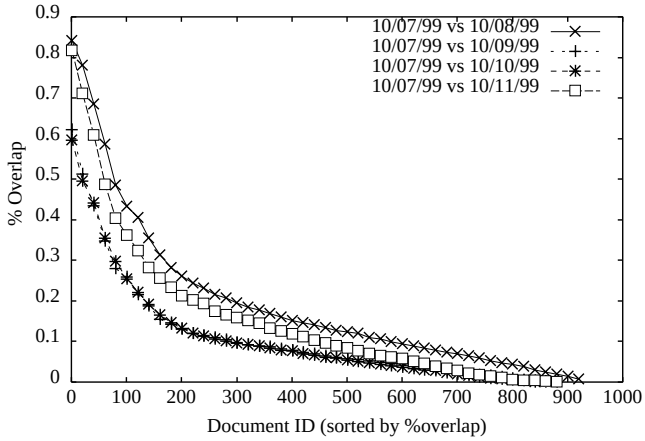


Figure 7: Stability of interest group.

Figure 7 shows the extent of overlap (as a percentage) for several pairs of days we studied. As we can see, the overlap is not very large. Only a few documents have over half of the domains making requests on both days. We observe similar results during other periods of traces as well. One explanation for this can be that the interest groups for the documents may not stabilize within a day, possibly because our definition of domains is too fine-grained. Another explanation could be that domain-level proxy caching can reduce the likelihood of multiple requests for a Web page emanating from a domain. As part of our future work, we plan to investigate why there is a large variation in the set of domains that request a Web page, from one day to another.

5.3 Spatial Locality

In this section, we investigate whether the domain membership is significant, i.e., whether clients belonging to the

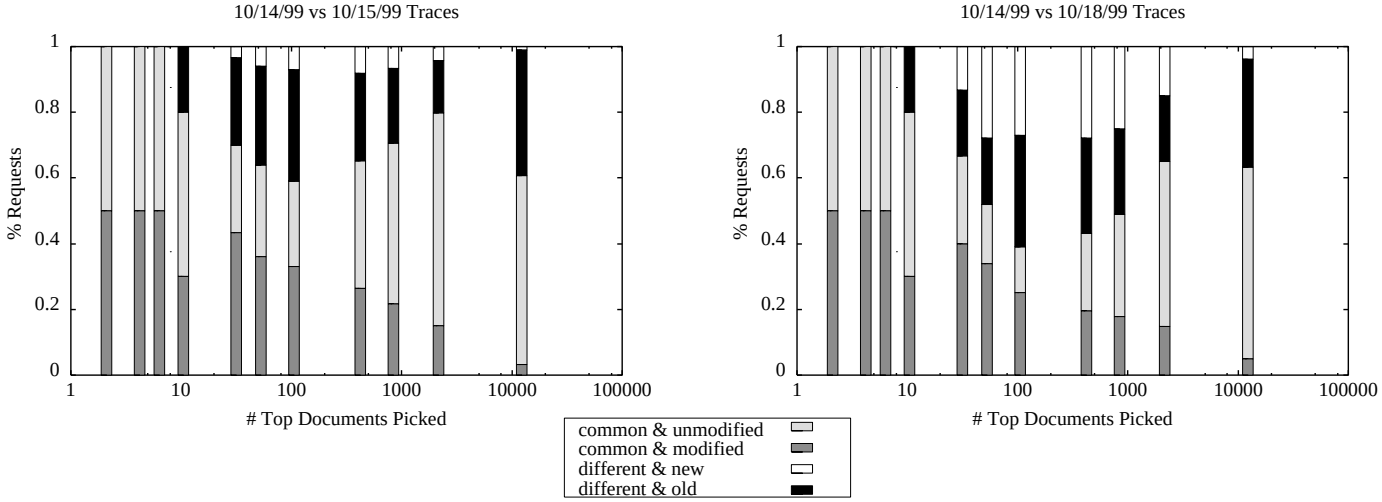


Figure 6: Stability of document popularity.

same domain are more likely to share requests than clients picked at random. This kind of spatial locality has obvious implications for performance, particularly with respect to the effectiveness of proxy caching.

Our approach, based on that of [29], is to compute the degree of local sharing (i.e., intra-domain sharing) and non-local sharing (i.e., inter-domain sharing) under the following two situations: (i) when clients are assigned to domains based on their DNS names (*true* assignment), and (ii) when clients are assigned to domains randomly, while preserving the size of each domain as before (*random* assignment). A major distinction between our study and [29] is that our domains correspond to (often) large and diverse DNS level-two domains whereas [29] defines "domains" to be the rather small and much less diverse departments within a single university.

The top two graphs in Figure 8 show the intra-domain and inter-domain sharing on 12/17/98, and the lower two graphs show the results for 10/7/99. In both cases, the inter-domain sharing with random assignment is comparable to that with true assignment, as we would expect. In contrast, intra-domain sharing with true assignment is noticeably higher than with random assignment for 10/7/99. This is also observed in the traces of many other periods. On the other hand, the intra-domain sharing on 12/17/98 (the day of Operation Desert Fox) using true assignment is comparable to using random assignment.

From these results, we conclude the following. In most cases domain membership is significant, i.e., clients belonging to the same domain are more likely to share requests than clients picked at random. This is in agreement with the findings reported in [29] despite our domains being much larger and more diverse. However, when there is a "hot" event, the global interest can become so dominant that even clients picked at random tend to share many requests, thereby diminishing the significance of domain membership.

5.4 The Applicability of Zipf's Law to Web Requests

Several prior studies have investigated the applicability of Zipf's law to Web accesses. The majority of studies have been based on proxy traces. These studies have concluded that Web requests follows a Zipf-like distribution, C/i^{α}, with α close to or smaller than 1 [12, 7, 23, 5]. There have been a few studies based on traces from relatively small servers. For instance, [2] showed that the distribution of accesses seen by a Web server follows Zipf's law (i.e., $\alpha = 1$).

We investigate this issue further by studying access logs from both the server and the proxy. We plot the number of document accesses versus document ranking on log-log scale. Due to space limitation, we only show the plots for the server traces (Figure 9).

We make the following observations. The curves for both the server traces and the proxy traces (not shown) fit a straight line reasonably well when ignoring the first 100 documents as in [5]. The few most popular documents' popularity deviates from the straight line especially in left of Figure 9. The remaining documents' popularity fits a straight line reasonably well. The straight line on the log-log scale implies that the request frequency is proportional to $1/i^{\alpha}$. The values of α are obtained using least square fitting, excluding the top 100 documents (as in [5]), and also excluding the flat tail. This shows that the finding of previous proxy-based studies, that Web accesses follow a Zipf-like distribution, extends to accesses seen by a server as well.

However, the range of α values corresponding to our server traces is consistently and significantly higher than that reported in previous proxy-based studies. Specifically, the value of α in the server traces is typically around 1.4 - 1.6, with lowest being 1.397 and highest being 1.816. In contrast, proxy-based studies have reported a much smaller α, typically less than 1.0.

Another interesting observation is that the value of α is the highest (1.816) on 12/17/98, when there was an unusual event of global interest (Operation Desert Fox). This is as we would expect. During such period users from all over

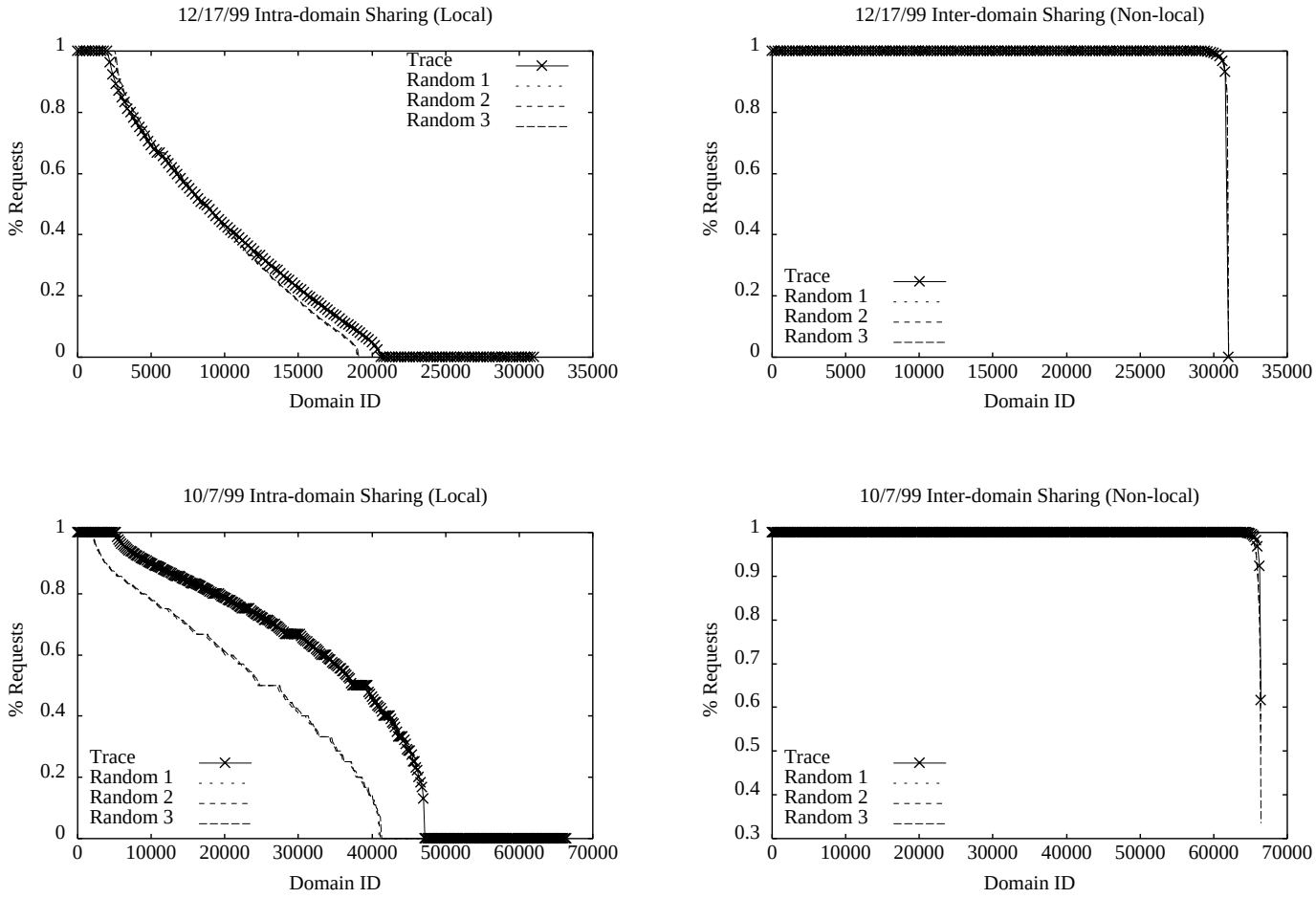


Figure 8: Compare the intra-domain and inter-domain sharing in the real traces with four random client to domain assignments.

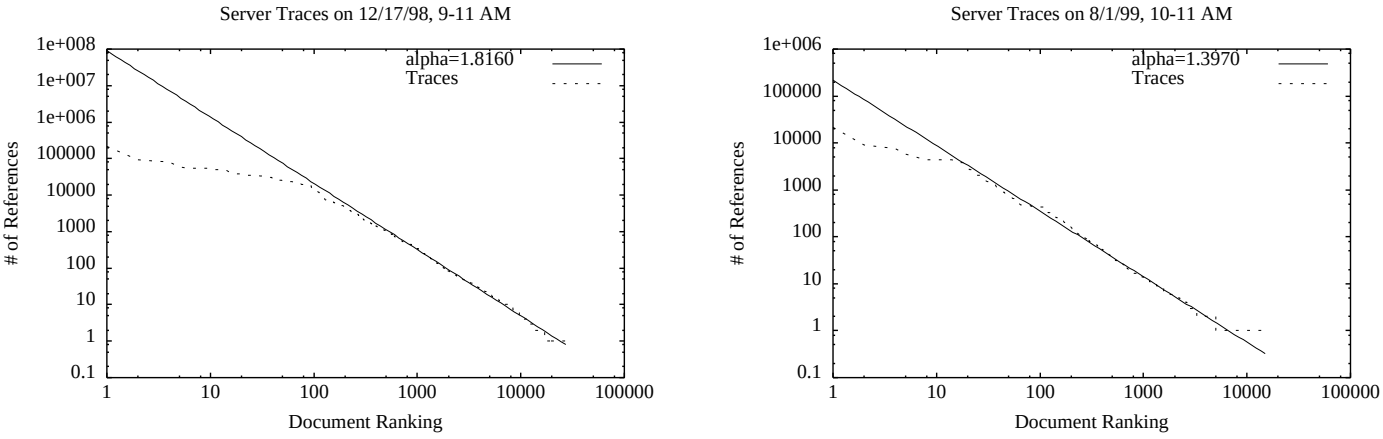


Figure 9: Frequency of document accesses versus document ranking (server traces).

the world are interested in a small set of pages related to the event, making hot documents extremely hot and cold documents colder than usual. So the difference in the hit count of hot pages and cold pages is enlarged, making α larger.

To further study the impact of different α values, we also plot the cumulative distribution of requests to popular documents. Figure 10 shows the cumulative probability of access for the top r% of documents for the server traces from 12/17/98 (the highest α due to flash crowds) and 8/5/98 (a "normal" day), and the Microsoft proxy traces from 10/6/99. From the server traces, we observe that on 12/17/98 the top 2% of the documents account for 90% of the accesses, and on 8/5/98 the top 4% of the documents account for 90% of the accesses. The latter is also observed for the other days of the server traces. In comparison, it takes 36% of the documents to account for 90% of the accesses in the proxy traces, which is also shown in [5]; and it takes 10% of the documents to account for 90% of the accesses in the traces from less popular servers collected in 1994 and 1995 [3]. Hence it is evident that Web accesses seen at the server tend to be more concentrated than those seen at the proxies. Moreover the accesses seen at the popular MSNBC server site tend to be more concentrated than those seen at less popular servers. This implies that techniques such as reverse caching and replication could go a long way in alleviating server load, especially for popular servers.

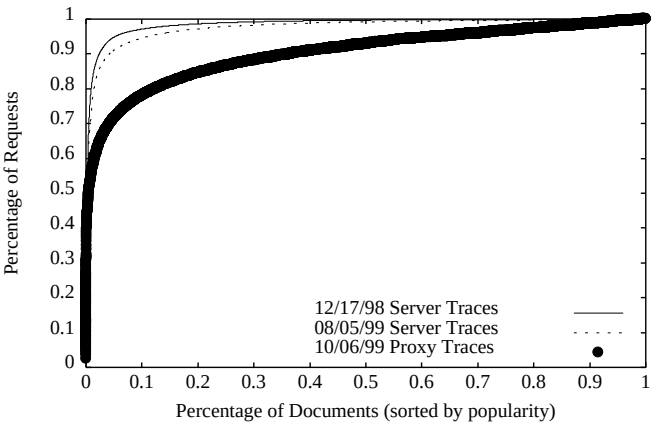


Figure 10: Cumulative distribution of requests to documents.

5.4.1 Reason and Implication of Larger α at Web servers

We have observed that the α values in the server traces are consistently and significantly higher than those in the proxy traces. Although our access logs are deficient in that they do not contain accesses to images, such deficiency is unlikely to consistently bias α towards a higher value. As found in [28], the α value for accesses to image and HTML files (based on a large university proxy trace) are 0.80 and 0.76, respectively. This suggests excluding the accesses to images are unlikely to bias α towards a higher value. The following straightforward calculation sheds more light on this.

Suppose, when we exclude image files, two HTML files receive W and W' accesses each. The ranking of the two are R and R' respectively. Then we have $\alpha = \frac{log(W)-log(W')}{log(R)-log(R')}$. After including image files, we have $\alpha = \frac{log(W)-log(W')}{log(k+i+R)-log(k+R')}$, where $k \geq 0$, and $i \geq 0$. Adding k to the original ranking of both objects is because including more files can move the original files' ranking behind (e.g., once image files are included, a file with a popularity rank of 100 can have its rank pushed back to say 200). Adding i to one of the object is because adding more files can possibly enlarge the difference in ranking of two files. Now we make the following observations:

- If $i = 0$, α would increase after including image files.
- If i is large enough to offset k, α would decrease after including image files.

Based both on analysis and previous trace-based studies, we conclude that the lack of image access information should not create a systematic bias towards higher α values. Rather we believe that the higher α values we observe is because we have studied accesses seen at servers rather than at proxies. A number of previous studies [5, 29] have shown that popular Web pages are spread almost evenly across hot Web servers. This implies that a proxy aggregating requests to a number of popular servers would have a slower decay in popularity than any individual server. This is illustrated by the following simplified calculation.

Suppose a proxy accesses s Web servers. The Web server i has n_i documents, with decay coefficient α_i. In addition, it receives A_i accesses to its most popular Web page. Then according to the Zipf-like distribution, we have

$$log(A_i) = \alpha_i * log(n_i)$$

The decay coefficient at the proxy can be computed as:

$$\alpha_{proxy} = \frac{log(max(A_1, A_2, ..., A_s))}{log(n_1 + n_2 + ... + n_s)}$$

For a simple case when $A_1 = A_2 = ... = A_s = A$, $n_1 = n_2 = ... = n_s = n$, and $\alpha_1 = \alpha_2 = ... = \alpha_s = \alpha_{server}$, we have

$$\alpha_{proxy} = \frac{log(A)}{log(s * n)} = \alpha_{server} * \frac{log(n)}{log(s * n)} < \alpha_{server}$$

The exact difference between α_{proxy} and α_{server} depends on s (the total number of popular servers). Of course, things are much more complicated in practice: (i) the Web accesses do not strictly follow a Zipf-like distribution, especially for the most popular documents; (ii) Web servers are very heterogeneous. It is very hard to compute α exactly. On the other hand, as long as it is true that the popular Web pages are spread almost evenly across hot Web servers, the α at the proxy is lower than at each individual popular Web server.

In summary, we observe the access patterns at both the server and the proxy exhibit a Zipf-like distribution. The value of α is much higher for the server accesses than for the proxy. In some cases, the top 2% documents account for 90% accesses. In contrast, the accesses seen at the proxy traces are more heterogeneous. They take up to 39% to account for 90% accesses. This suggests that the hit rate for reverse caching or replication is likely to be significantly higher than that observed with traditional proxy caching, and that caching or replicating a small set of files would cut down server load significantly.

6. CONCLUSIONS

In this paper, we have studied the dynamics of a large commercial Web site, which is consistently ranked among the busiest in the Internet. We have analyzed the dynamics of both the content and the client accesses at this site.

6.1 Summary of Key Results

Our main findings are:

1. The server content tends to be highly dynamic with around 6000 files created and 24000 modified over a one-week period. For the subset of files that are modified, the duration between successive modifications tends to lie between an hour and 24 hours (i.e., a day).
2. Past modification behavior of a file, if averaged over a sufficient number of samples, yield a rough predictor of future modification behavior. The estimation error is within a factor of two 65% of the time and within a factor of five 90% of the time.
3. The *vdelta* between successive versions of files is often extremely small, implying that modifications made to the (HTML) content tend to be minor.
4. The popularity of files tends to be stable over a timescale of days. Of the 100 most popular documents on a given day, 60–100% remain among the top 100 for up to 5 days. However, the set of domains from which the popular files are accessed tends to change significantly from day to day. For example, there is only a 40% overlap when considering the top 100 files. This may be a consequence of our (fine-grained) definition of a domain, and we are currently exploring this further.
5. Organizational (i.e., domain) membership of clients tends to have a significant (positive) impact on the degree of local sharing. This agrees with the results reported in [29]. However our study differs from [29] in several important ways: (i) [29] study the accesses made by a university, while we study the accesses to a large commercial news site from all over the world. The clients in our traces are thus more heterogeneous; (ii) Our domains correspond to (often) large and diverse DNS level-two domains whereas [29] defines "domains" to be the rather small and much less diverse departments within a single university; (iii) We also find when there is a globally-interesting event (such as Operation Desert Fox in December 1998) that cuts across organizational boundaries, the organizational membership is no longer significant in affecting the users' accesses pattern.
6. File popularity tends to follow a Zipf-like distribution. However, the parameter α seen at the server is in the range 1.4–1.6, which is much larger than has been reported in the literature (based on the analysis of proxy logs). The large value of α implies, for instance, that just the top 2% of documents could account for 90% of the accesses.
7. The popularity of most documents tends to drop off with age. However, some documents tend to maintain their popularity for a significant length of time.
8. The majority of first-time accesses from a domain are to unpopular documents at least a day old.

6.2 Broad Implications

Our study of both server content and access dynamics has broad implications for the future evolution of the Web.

6.2.1 Implications for Cache Consistency Algorithms

We find that files are modified frequently (Section 4.2) and that a large part of a user's requests are to modified files (Section 5.1.2). For instance, over half of the repeated accesses (accesses to files previously requested by the same domain) are to a modified file. This pattern of file modification and access underscores the importance of having efficient cache consistency control mechanisms (e.g., [6] [31]). The modification history of a file can yield a rough predictor (Section 4.3) for adaptive TTL based schemes (e.g., [14]).

Second, the *vdelta* between successive versions of files is often extremely small (Section 4.4), so techniques such as delta encoding [20] will be very effective.

6.2.2 Implications for Prefetching and "Push"

Our study has several implications for prefetching and server-based "push".

First, file creation tends to be a frequent event (Section 4.1). When a popular news story is updated, it may be assigned a new file name. The absence of past access history makes the task of prefetching or preemptively pushing out newly created content challenging. Clever techniques, such as using the access history of the surrounding pages based on Web linkage structure to make predictions, may be needed.

Second, the high frequency of file modification and accesses to modified files suggests that prefetching or pushing such files would be very useful. Moreover, since the popular objects stay popular for a relatively long period of time (Section 5.2.1), it makes sense to prefetch (or push) previously popular files that have undergone modification.

Lastly, the domain membership of clients is of significance. Clients belonging to the same domain are more likely to share requests than clients picked at random (Section 5.3). On the other hand, the stability of interest groups is not very high (Section 5.2.2), so it may be challenging to have server push documents discriminatively to the clients. In future work, we plan to investigate this issue further.

6.2.3 Implications for Web Caching

The implications of our findings for Web caching are twofold.

On the positive side, we find the accesses seen by the Web server are more concentrated to a small set of documents than the previous proxy-based studies, with a higher α (Section 5.4). This implies caching (or replicating) a small number of popular documents may potentially reduce server load and/or Web traffic significantly.

On the negative side, it appears that first-time misses, though as high as 40% of the all accesses [27], would be hard to cut down significantly. This is because most first-time accesses tend to be to old and unpopular documents (Section 5.1.2), so it is very hard to accurately predict such accesses. However, prefetching or pushing modified objects to users is still

beneficial due to the high frequency of modification rate and access rate. Therefore, we believe designing efficient cache consistency algorithms, and making more (hitherto uncachable) objects cachable are two major directions to improve the efficiency of Web caching. Replication done under server control may be a viable solution to both issues.

7. ONGOING AND FUTURE WORK

We are presently analyzing a larger content log set, investigating better heuristics for identifying client domains, and examining why the stability in interest group is low. In the longer term, we plan to study data sets from other large server sites to confirm or refute the findings reported here. We also plan to develop efficient cache consistency algorithms based on the insights we have gained from our work.

8. ACKNOWLEDGMENTS

Jason Bender and Ian Marriott made the trace data available to us and were very patient with our many questions. Erich Nahum offered several valuable suggestions on revising this paper as did the anonymous SIGCOMM reviewers. Kiem-Phong Vo provided us with a binary executable of the *vdelta* program. Damon Cole, Susan Dumais, Niccole Golden, Chris Haslam, Eric Horvitz, and Geoff Voelker helped us in various ways. We would like to thank them all.

9. REFERENCES

[1] http://www.abcnews.com.

[2] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira. Characterizing Reference Locality in the WWW. In *Proc. of PDIS'96*, December 1996.

[3] M. F. Arlitt and C. L. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proc. of SIGMETRICS'96*, May 1996.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP Behavior of a Busy Web Server: Analysis and Improvements. In *Proc. Infocom 1998*, March 1998.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of INFOCOMM'99*, March 1999.

[6] E. Cohen, B. Krishnamurthy, and J. Rexford. Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters. In *Proc. SIGCOMM'98*, September 1998.

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW client-based traces. *Technical Report TR-95-010, Boston University, Computer Science Dept., USA*, April 1995.

[8] http://www.cnn.com.

[9] F. Douglis, A. Feldman, B. Krishnamurthy, and J. C. Mogul. Rate of Change and Other Metrics: A Live Study of the World Wide Web. In *Proc. USITS '97*, December 1997.

[10] B. M. Duska, D. Marwood, and M. J. Feeley. The Measured Access of World Wide Web Proxy Caches. In *Proc. USITS '97*, December 1997.

[11] L. Fan, Q. Jacobson, P. Cao, and W. Lin. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In *Proc. of SIGMETRICS'99*, May 1999.

[12] S. Glassman. A Caching Relay for the World Wide Web. In *First International Conference on the World Wide Web*, CERN, Geneva, Switzerland, May 1994.

[13] S. D. Gribble and E. A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. In *Proc. USITS '97*, December 1997.

[14] J. Gwertzman, M. Seltzer. World-Wide Web Cache Consistency. In *Proc. USENIX '96*, January 1996.

[15] J. Hunt, K. P. Vo, and W. Tichy. Delta Algorithms: An Empirical Analysis. In *ACM Transactions on Software Engineering and Methodology*, Vol. 7, 1998.

[16] W. LeFebvre and K. Craig. Rapid Reverse DNS Lookups for Web Servers. In *Proc. USITS '99*, October 1999.

[17] C. Liu and P. Cao. Maintaining Strong Cache Consistency in the World-Wide Web. In *Proc. ICDCS'97*, pp. 12-21, May 1997.

[18] Microsoft Corporation. http://www.microsoft.com.

[19] J. C. Mogul. Network Behavior of a Busy Web Server and its Clients. *Research Report 95/5, Compaq Western Research Lab*, October 1995.

[20] J. C. Mogul, F. Douglis, A. Feldman, and B. Krishnamurthy. Potential Benefits of Delta Encoding and Data Compression for HTTP. In *Proc. SIGCOMM '97*, September 1997.

[21] http://www.mediametrix.com.

[22] http://www.msnbc.com.

[23] N. Nishikawa, T. Hosokawa, Y. Mori, K. Yoshidab, and H. Tsujia. Memory-based Architecture for Distributed WWW Caching Proxy. In *7th WWW Conference*, April 1998.

[24] V. N. Padmanabhan and J. C. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. *ACM SIGCOMM Computer Communication Review*, July 1996.

[25] V. N. Padmanabhan and L. Qiu. "The Content and Access Dynamics of a Busy Web Server: Findings and Implications". *Microsoft Research Technical Report MSR-TR-2000-13*, February 2000.

[26] J. Touch. The LSAM Proxy Cache - a Multicast Distributed Virtual Cache.

[27] A. Vahdat, M. Dahlin, T. Anderson, and A. Aggarwal, Active Names: Flexible Location and Transport of Wide-Area Resources. In *Proc. USITS '99*, October 1999.

[28] G. M. Voelker. Personal Communication, Feb 2000.

[29] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, H. Levy. Organization-based Analysis of Web-Object Sharing and Caching. In *Proc. USITS '99*, October 1999.

[30] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, H. Levy. On the Scale and Performance of Cooperative Web Proxy Caching. In *Proc. SOSP '99*, December 1999.

[31] H. Yu, L. Breslau, and S. Shenker. A Scalable Web Cache Consistency Architecture. In *Proc. of SIGCOMM'99*, September 1999.