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

Venkata N. Padmanabhan
padmanab@microsoft.com
Microsoft Research

Lili Qiu
lqiu@cs.cornell.edu
Cornell University

February 2000

Technical Report
MSR-TR-2000-13

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

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

Venkata N. Padmanabhan
padmanab@microsoft.com
Microsoft Research

Lili Qiu
lqiu@cs.cornell.edu
Cornell University

February 2000

Abstract

In this paper, we study the dynamics of 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 the Internet bandwidth consumption caused by the web, as well as improving user-perceived latency. Techniques such as Web caching and prefetching or server-based "push" of Web content have been widely studied, and several new algorithms and protocols for these have been proposed. We believe understanding the characteristics of the Web workload and traffic are very important for the design and evaluation of these algorithms. In this paper, we study the dynamics of one of the busiest Web sites in the Internet today, and discuss the 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, and proxies. The servers are typically the originators of content while the clients are the consumers of content. Proxies, when present, mediate the communication between a set of clients and a subset/all of the servers. As such, each of these three 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 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 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 a large commercial server site, which, to preserve anonymity for SIGCOMM review, we will refer to simply as *FooBar*. FooBar is a typical large commercial news site in the same category as the likes of CNN [8], MSNBC [23], and ABCNews [1], and is consistently ranked among the busiest sites in the Web [22]. 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 FooBar 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 [21]. 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 FooBar site generalize to the other large sites in the Internet. While there may certainly be characteristics that are peculiar to the FooBar site, we believe that the operation of the site is quite typical of large news sites such as CNN, MSNBC, and ABCNews. 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 is organized as follows. In Section 2, we survey previous work. We present a discussion of the architecture of the FooBar 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 upon ongoing and future work.

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 [28] [29] 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 picked at random. 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 FooBar 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 FooBar site back-end in our study, so the possibility of missed updates is diminished/eliminated.

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 [20] [4]. Others have focussed on very specific aspects of the server's operation (e.g., [18] 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 FooBar server cluster sees, on average, over 25 million accesses each day to the 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 FooBar 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 FooBar server site, and discuss the trace data that we gathered and processed.

3.1 Server Site Architecture

The FooBar server site comprises a cluster of over 40 server nodes, each running the Microsoft Internet Information Server (IIS) [19]. 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 FooBar site. Second, within each sub-cluster that shares a VIP address, the Windows Load Balancing Service (WLBS) [19] 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 any significant way since our analysis of access dynamics focuses on Web *pages* (as defined by the HTML content) rather than on the individual files.

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 [1] were especially interesting, because these corresponded to a "hot" news event, namely the launching of *Operation Desert Fox* by the US military against Iraq.

The FooBar server site saw, on average, over 25 million client accesses for 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 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.

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

	12/17 - 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	all day	all day
% total 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
% Domain found[2]	58.587	-	76.227	78.967	78.344	-
Domains	41025	-	75569	117615	396528	-
% GET	99.818	99.050	99.065	99.065	99.008	99.059
% POST	0.088	0.448	0.464	0.474	0.512	0.475
% HEAD	0.082	0.406	0.392	0.327	0.350	0.336
% Other methods	0.012	0.096	0.079	0.134	0.130	0.130
% status=200	58.084	55.913	57.104	56.195	55.088	54.744
% status=302	4.554	15.017	15.267	17.647	16.047	18.443
% status=304	36.946	27.529	26.231	23.812	26.517	24.501
% status=400	0.010	0.023	0.025	0.031	0.029	0.026
% status=403	0.003	0.024	0.018	0.013	0.015	0.018
% status=404	0.327	1.347	1.241	1.738	1.654	1.661
% status=500	0.012	0.089	0.070	0.131	0.125	0.126
% Other status	0.064	0.058	0.044	0.433	0.525	0.481

Table 1: Overall trace statistics

3.3 Content Creation and Modification Logs

The back-end of the FooBar site uses the Microsoft Content Replication System (CRS) [19] 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 [3]. 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 FooBar, we were able to obtain a log of 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 [4]. The content log, although limited in scope, enables us to get some insight into the evolution of files as they undergo modification.

[3] 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".

[4] 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.

3.5 Proxy Logs

In some of our analyses (Section 5.4), we compare the access characteristics of the FooBar server with that of a busy caching proxy. We obtained logs from a proxy cache that serves a large campus population with 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 implication 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. Figure 1(a) shows the number of file creation and modification events (computed hourly) over a one-week period (midnight Saturday, 10/9/99 through midnight Friday, 10/15/99 local time). Not surprisingly there is a clear diurnal cycle in the file creation and modification process. There is a trough at nighttime, and several peaks during the daytime. We believe the reason for these peaks is that even if the actual content generation/modification process is spread out uniformly, the CRS replication process is only scheduled to run from time to time (i.e., it replicates a bunch of files at a time rather than individual files; the only exception is a "hot" file that needs to be updated immediately). The time of replication is what really matters because only after it is replicated that the new content becomes available to clients.

A weekly cycle is also evident from Figure 1(a). There tend to be fewer events on the weekend (the first two days shown in the figure) than during the week.

In Table 2, we tabulate a more detailed breakdown of the event counts. 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" [25]. 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., [7, 30]).

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 [28], 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.

4.2 Distribution of Modification Intervals

Next, we turn to Figure 1(b), 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 [15].

Figure 2(a) shows a scatter plot of pairs of successive modification intervals for a file. If duration of the previous modification interval were indicative of the next one, we would have expected the points to be clustered along a positively-sloped line. It is clear from the figure that this is not the case. There are several instances where a short modification interval for a file is followed immediately by a long one, and vice versa. Indeed, the coefficient of correlation [16] is only 0.23: mildly positive.

We then explored the possibility of using a larger

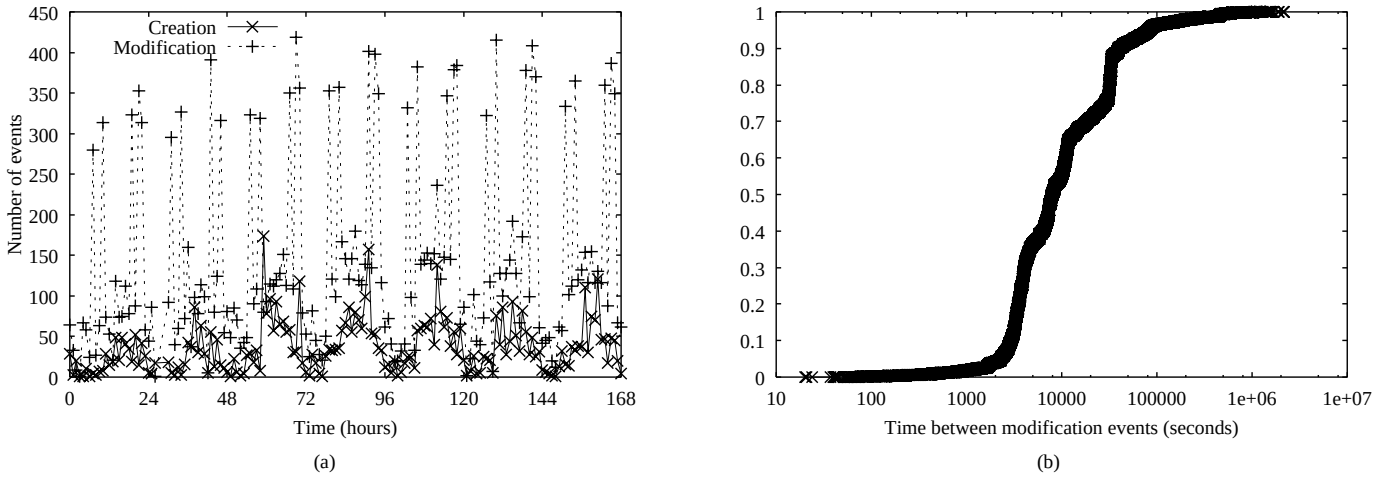


Figure 1: (a) An hourly count of file creation and modification events over a one-week period (Saturday through Friday). (b) CDF of the time interval between successive modifications of a file (conditioned on the file being modified).

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.

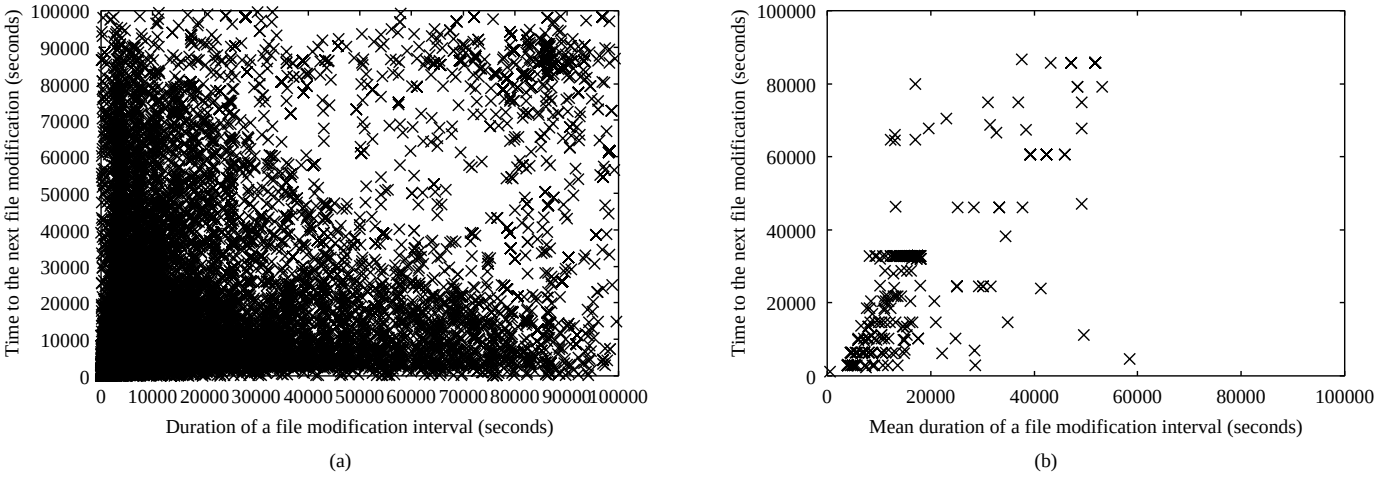


Figure 2: (a) A scatter plot where a point (x, y) represents two successive modification intervals for a file, the first of duration x and the second of duration y. (b) A scatter plot of points (x, y) where x is the mean duration of the modification interval for a file (computed using past history), and y is the duration of a new modification interval for that file.

amount of history than just a single piece of information, i.e., the duration of the previous modification interval. We divided the 4-week duration of our CRS logs into roughly equal halves. Using the data from the first half, we determined the mean modification interval for each file. In the end, we only retained the set of files for which the mean was computed over 10 or more samples (i.e., there was "sufficient" modification history). Then using the second half of the log data, we determined the length of the first modification interval for each of the files in the set. The question we wanted to answer was how good a predictor the mean was of the duration of the new modification interval. Figure 2(b) shows the scatter plot of these two quantities. We observe that there is a fairly strong positive correlation between the mean duration of the modification interval from the past and the duration of the new interval. The coefficient of correlation 0.79 confirms this.

Thus, we believe it may be appropriate to use the modification interval from the past as a predictor of the future so long as a sufficient number of samples from the past are averaged. In the context of Web cache consistency, this suggests that an adaptive TTL scheme, that uses past modification behavior of a file to adapt future TTL settings on the file [15], would work well provided there is an invalidation mechanism (e.g., callbacks) available as a backup.

4.4 Extent of Change upon File Modification

We now examine just how much a file changes when it is modified. First, we consider the change in file size. Figure 3(a) shows the CDF for the change in file size (unsigned magnitude) in terms of bytes. We observe that there is little change in file size despite the modification. Over 70% of the modifications cause less than a 1% change in the file size (due to space limitations, we have not shown the graph of file size change expressed as a percentage). Although it is certainly possible for the content to change significantly while maintaining the same file size, intuitively it seems too coincidental to be likely. The analysis based on content logs we present next sheds more light on this. (Despite the ambiguity associated with inferring the extent of change in file content from the change in file size, we analyzed the latter as well because we had file size information for a lot more files than we had content logs for.)

We used the content logs (which, as described in Section 3.4, were available for a small subset of the HTML content) to explore more carefully how much a file changes upon modification. Since we were examining just HTML content, we decided to focus on the *visible* textual content, i.e., text that would be displayed by a client browser. To this end, we extracted the visible text by stripping out the HTML tags from each version of a file. We then quantified how similar two successive versions of a file were using the *cosine similarity metric* [14]. This is a standard metric used in information retrieval. For each document, a term (i.e., word) frequency vector is constructed. The cosine similarity metric is then just the inner-product of the two vectors divided by the product of their lengths, i.e., the cosine of the angle between the vectors. The more similar the files are, the closer the metric is to 1. The similarity metric is simple in that it only considers word frequency without caring about the order in which the words appear. To prevent "insignificant" words such as "and", "the", etc. from overwhelming the more significant ones, we created a list of common words to disregard in the similarity computation. However it is not perfect, since it does not consider the order of words.

Figure 3(b) shows the CDF of the text similarity metric computed over successive versions of files. We see that the metric tends to be close to 1, which indicates that little changes in terms of the visible textual content. On closer examination, we found (at least) a couple of common modes of (minor) textual change: (a) a date/time

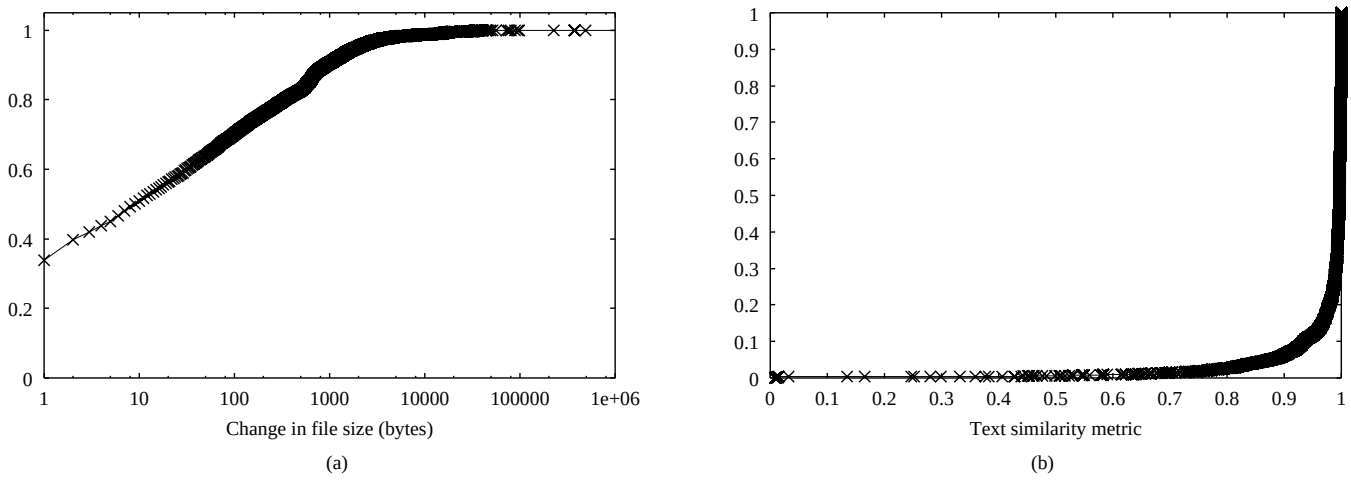


Figure 3: (a) The CDF of change in file size (in bytes) between successive versions. (b) The CDF of the text similarity metric computed over successive versions of files.

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.

In conclusion, our analysis suggests that successive versions of files are very similar, both in terms of size and in terms of content. This implies that techniques such as delta-encoding [21] 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 good 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 FooBar server site. Access dynamics is crucial to the effectiveness of Web caching. In particular, insights to the following issues may significantly affect the way 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 and its implication for the effectiveness of Web caching in reducing Internet traffic

As shown in Table 1, over 99% of the requests employ the GET method. Moreover, among all the requests, the ones with HTTP response status code 200 (action successful) account for 55%. Around 15% to 18.5% of the requests have response status code 302 (moved temporarily), and around 23% to 37% have response status code 304 (not modified). The latter implies that the poten-

tial 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.

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 document 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 increasing order by the total number of accesses. It is evident from the graphs that most documents receive a lot more accesses soon after their creation than afterwards. On the other hand, there are a number of documents (the ones denoted with a large document ID) that remain hot for the entire five-day period under study. These files are index pages, such as the default front page for the FooBar site.

5.1.2 Classification of accesses according to file modification/creation

Previous research [26] 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 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.

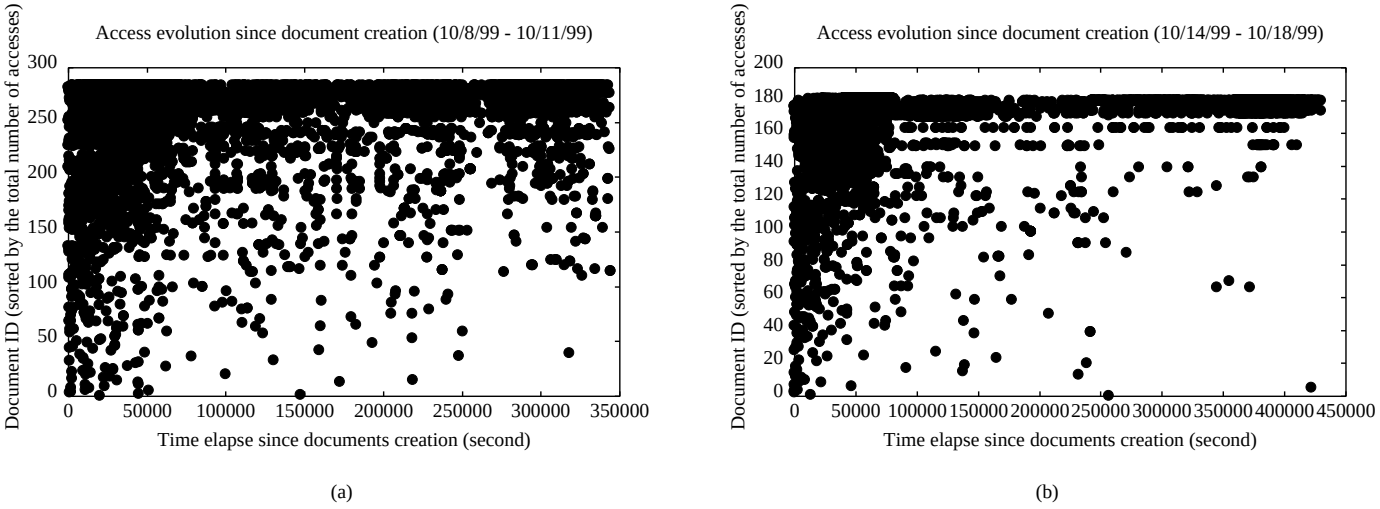
We also classify repeated accesses to objects according to their modification history. Specifically, we divide the accesses into two groups: accesses to objects that are modified on the day of access, and those that are not. As shown in Table 3 (columns 4 and 5), over half of the repeated accesses are to the modified objects. This again suggests that designing a good cache consistency control mechanism will be very useful.

5.2 Temporal Stability of User Access

In this section, we analyze the temporal stability of Web accesses. In particular, the following questions are especially important to understand:

- How much does the popularity of Web pages vary 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?

OB 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.



(a) (b)

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

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

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

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 compare the overlap in the most popular pages selected from one day to the next. Our results are illustrated in Figure 5. X-axis is the number of most popular documents picked (e.g. $x = 10$ means we pick the 10 most popular documents), and 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 the web accesses in the same short time-frame (within 1 week period) are equally good for predicting which Web pages will be 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. The reason for this is that many of the very popular pages are *index* pages (such as the default front page for the FooBar 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

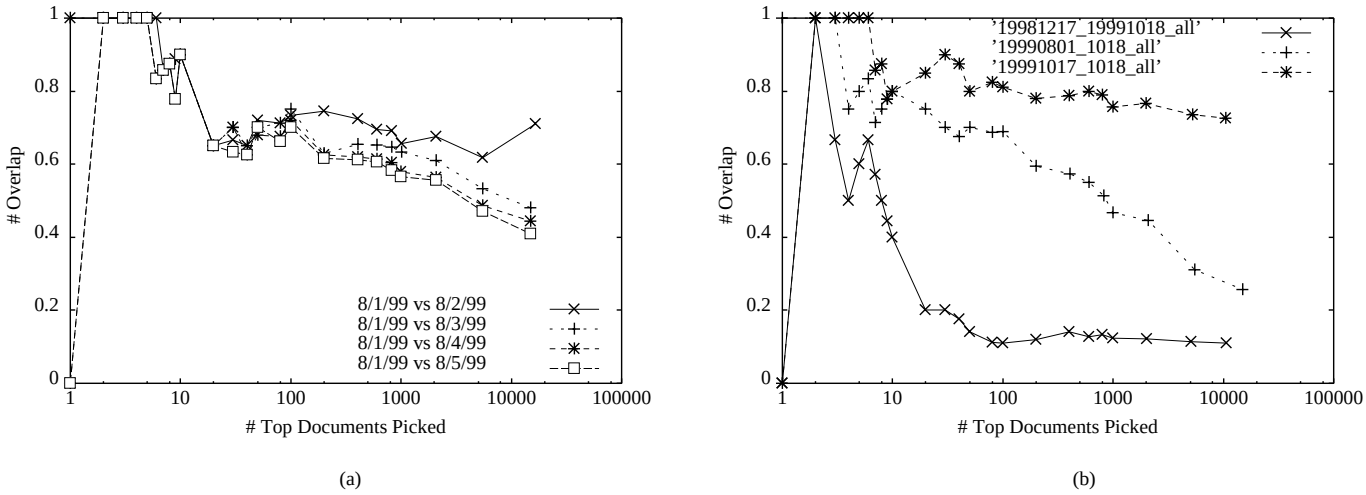


Figure 5: Stability of document popularity in both short and long period

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 are in existence 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 remain unmodified through both days. This implies that Web 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 the server. Bridging this gap will help reduce Internet traffic substantially as well as improving 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 Web pages that were in existence on both days. Even though the number of modified files is not very large, because of the frequent updates made to a small set of files (as observed in Section 4.1), designing good invalidation techniques for web documents is still very desirable.

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, although both counts are of the same order of magnitude.

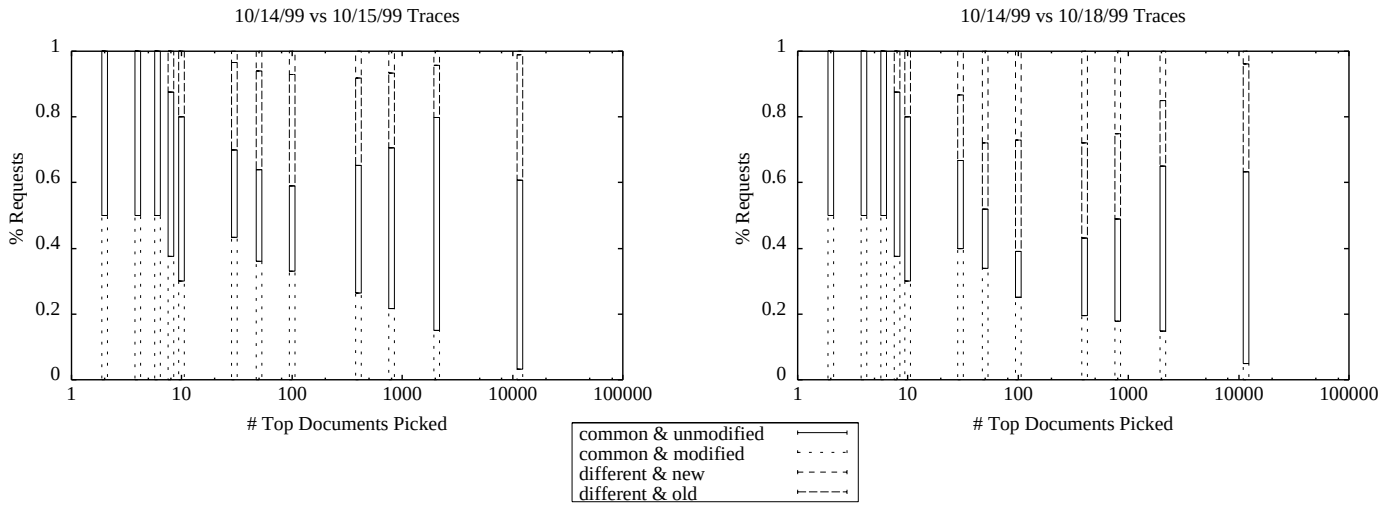


Figure 6: Stability of document popularity

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 every 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 access the page on each day, and determine the extent of overlap between these sets. As mentioned in Section 3.2, we ignore requests from clients for which the reverse DNS lookup fails. Since the percentage of failure is reasonably low (Table 1), it should not make a significant difference in our results.

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 that 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 are interested in understanding whether the domain membership is significant, that is, whether clients belonging to the 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 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) assign clients to domains based on their DNS names, and (ii) assign clients randomly to domains while preserving the size of each domain.

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 the 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

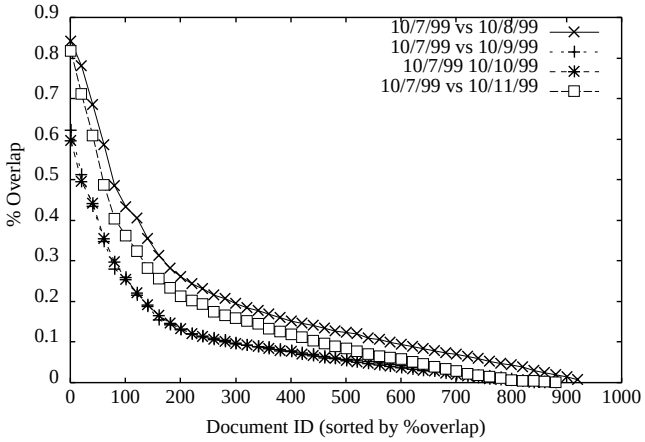


Figure 7: Stability of interest group

traces of many other periods. On the other hand, the intra-domain sharing on 12/17/98 (the day of Operation Desert Fox) is comparable with both true and 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 also conforms with the findings reported in [28]. 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, thus 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. For example, [5], one of the more recent pieces of work, gives a comprehensive summary of previous work on this issue. They report that the distribution of web requests from a fixed group of users follows a Zipf-like distribution, C/i^{α}, very well. The value of α varies from trace to trace, ranging from 0.64 to 0.83.

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.
- The value of α varies from trace to trace. The values of α in the server traces are consistently and significantly higher than those in the proxy traces. Specifically, the values of α in the server traces are mostly around 1.4 - 1.6, with lowest being 1.3970 and highest being 1.816. In comparison, the α in the proxy traces are much lower, around 1.0 for both the days we had proxy logs for. The consistently high α values of the web server is a very interesting phenomenon, and we discuss its cause

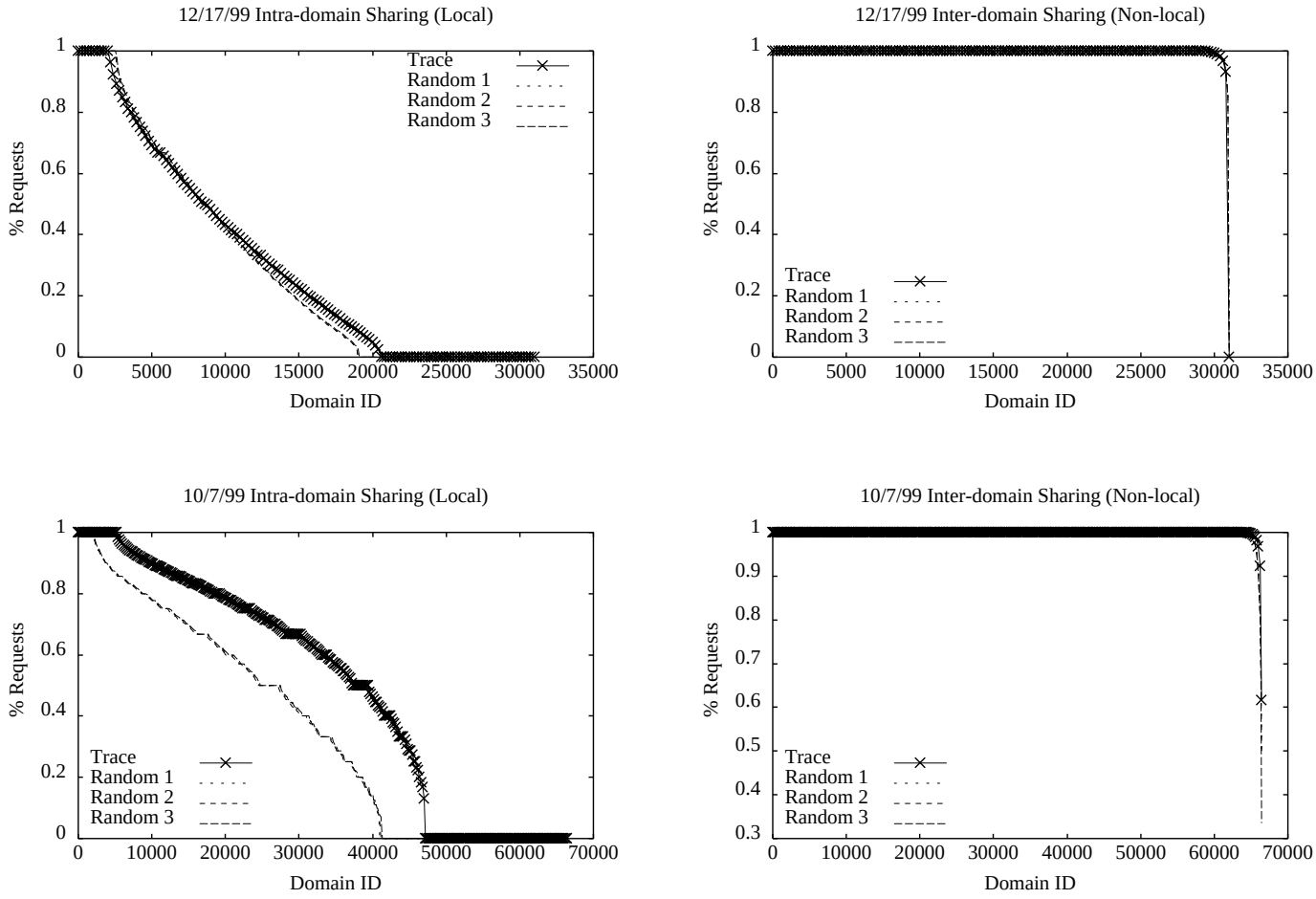


Figure 8: Compare the intra-domain and inter-domain sharing in the real traces with four random client to domain assignments

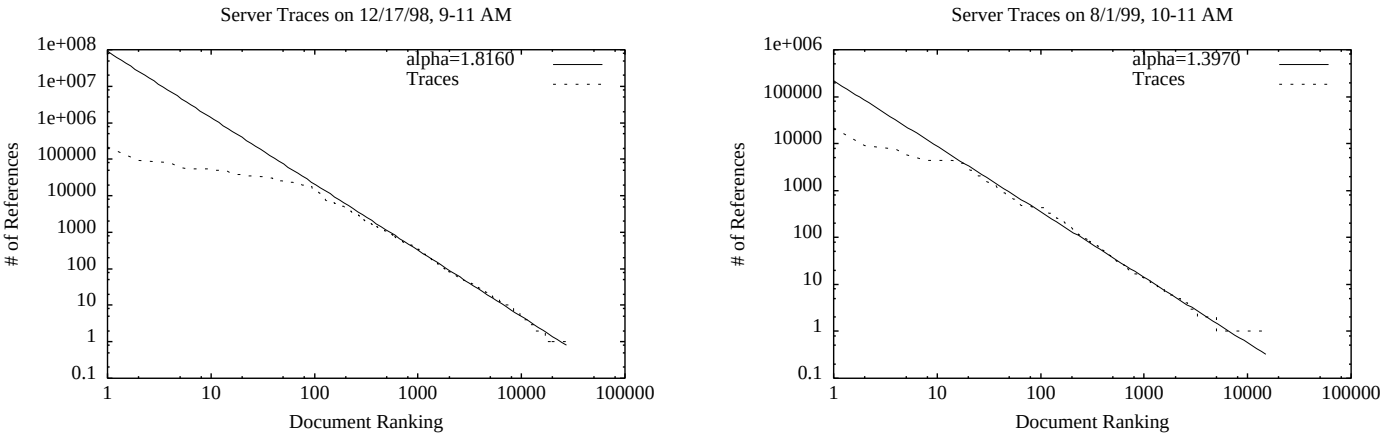


Figure 9: Frequency of document accesses versus document ranking (server traces)

and its implications in Section 5.4.1.

- The value of α is highest on 12/17/98, when there was an unusual global event interesting to people all over the world. This is as we would expect. Since during such period users all over 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, the difference in the hits count of hot pages and cold pages is thus enlarged, which contributes to a larger α value.

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 on 12/17/98 (the highest α) and the proxy traces on 10/6/99 (one of the most recent trace). As we can see, the top 2% documents account for 90% of accesses in the server traces. In contrast, it takes 36% to 39% of the documents to account for 90% of the accesses in the proxy traces. So, as [5] discovered, the 10/90 rule (i.e. 90% of the accesses go to 10% of the documents) does not apply for the web accesses seen at the proxies. On the other hand, according to our server traces, the web accesses observed at the server side are sometimes even more concentrated than the 10/90 rule. This implies that techniques such as reverse caching (i.e., placing a Web cache right in front of a server so that it is in a position to intercept all client requests) and replication could go a long way in alleviating server load.

5.4.1 Reason and Implication of Larger α at Web server

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 [27], 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 straight-forward 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. After including image files, a file ranked 100th popular can now become 200th popular). 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 $i = 0$, then α would increase after including image files.
- If i is large enough to offset k, then α would decrease after including image files.

From both analysis and other's trace study, we conclude that lacking image accesses should not consistently bias towards a higher α value. Rather we believe the higher α value is due to our study of server logs as opposed to proxy logs in the previous work. This can be explained as follows: A number of previous studies [5, 28] show popular web pages are spread almost evenly across hot web servers. This implies that the proxy aggregating requests to a number of popular servers should have a slower decay in popularity than any individual server. This can be illustrated by the following simplified example:

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

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

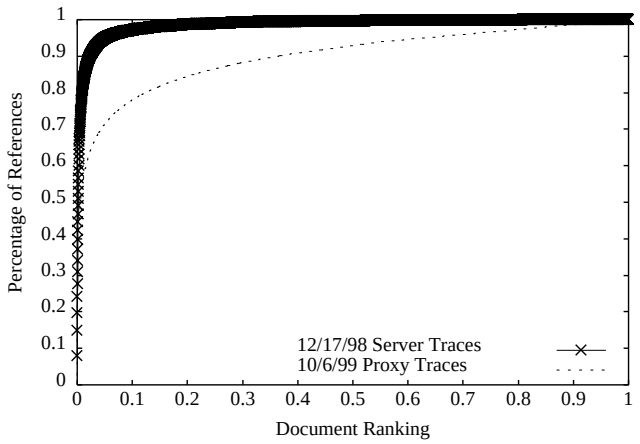


Figure 10: Cumulative distribution of requests to documents

The decay coefficient at the proxy can be computed as follows:

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

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 40% 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.

5.5 Access Pattern at Lower-level Proxies vs. Higher-level Proxies

Given the increasing widespread deployment of Web caching both at the edges and in the core of network, it is very useful to study how such deployment affects the web access pattern. In particular, we want to answer the following questions: (i) given the access pattern of the lower level proxies (or end-users), what is the access pattern at the higher- level proxies (HP)? (ii) If the access pattern at the lower level proxies exhibits a Zipf-like distribution, will the access pattern still have a Zipf-like distribution at the higher level? If so, what will the value of α at the higher level proxies be?

Answers to these questions depend on how lower-level proxies are assigned to higher-level proxies By assigning a lower-level proxy A to a higher-level proxy B, we mean whenever there is a cache miss at the lower-level proxy A, the request will be forwarded to the higher-level proxy B

for service. To make our analysis straight-forward, here we consider random assignments from lower-level proxies to higher-level proxies.

5.5.1 Analysis

We formulate the problem as follows:

Suppose without hierarchical caching, a page receives m accesses. When we employ a two-level caching hierarchy with x higher level proxies, shown in Figure 11, what is the average number of higher level proxies, z, that access the page?

To simplify our analysis, we ignore document invalidation, and also assume infinite cache size at all the proxies (both higher-level and lower-level proxies). Therefore multiple requests for a single document will be forwarded up towards the root only once by a higher-level proxy cache (basically, upon receipt of the request for the time from any of its children).

Based on these assumptions, we can derive the following formula:

$$z = \frac{x * m}{x + m - 1}$$

The detailed analysis is shown in Appendix A. Using the above result, we know if the ith popular page has C/i^{α} accesses from the lower level proxies, then it will have $C/(i^{\alpha} + \frac{C}{x})$ accesses from the higher level proxies (ignoring the -1 in the denominator since $x+m$ is likely to be much larger). When $\frac{C}{x}$ is much smaller than i^{α}, (or equivalently when i is large enough), then the access pattern at the higher level proxies looks very similar to the access pattern at the lower level. So the α values in both cases are close to each other for large i.

We validate the above analysis using our server traces. Our traces analysis matches the above derivation results, and show that the Zipf-like distribution also holds at the higher level proxies, and the value of α tends to be a little lower than at the lower level proxies.

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 files modified over a one-week period. For the subset of files that are modified, the time gap 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, tends to be a reasonably good predictor of future modification behavior.
3. Most (HTML) file modifications tend to be minor in terms of the change both in the file size and in the visible textual content.
4. File popularity tends to be distributed according to a Zipf-like distribution. However, the parameter α tends to be 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.
5. The popularity of files tends to be stable over a timescale of days. Of the top 100 documents in terms of popularity on a given day, 60–100% tend to remain among the top 100 for up to 5 days. However, the set of domains from which accesses to the popular documents are made tends to change significantly from day to day. For example, there is only a 40% overlap when considering the top

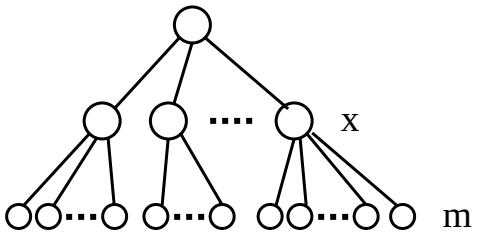


Figure 11: Two-level caching hierarchy

100 documents. This may be a consequence of our (fine-grained) definition of a domain, and we are currently exploring this further.

6. Organizational (i.e., domain) membership of clients tends to have a significant (positive) impact on the degree of local sharing, unless there is a globally-interesting event (such as Operation Desert Fox in December 1998) that cuts across organizational boundaries.
7. In the case of most popular documents, their popularity tends to drop off with age. However, some documents tend to maintain their popularity for a significant length of time.
8. Majority of first-time accesses (i.e., the first access to a document by any client in a domain) are to documents that are at least a day or more old and are unpopular.

6.2 Broad Implications

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

6.2.1 Implications for Cache Consistency Control Algorithms

We find the files are modified very frequently (in Section 4.2), and a large part of user's requests are to the modified objects (in Section 5.1.2). In particular, over half of the repeated accesses (accesses made to previously requested objects by the same domain) are to a modified object. The high frequency of modification rate and access rate underscore the importance of having efficient cache consistency control mechanisms (e.g., [7] [30]).

Second, the degree to which files are modified tends to be small (in Section 4.4), so techniques such as delta encoding [21] appear promising.

6.2.2 Implications for Prefetching or Server-based "Push"

Our study has the following implications on the prefetching or server-based "push":

- File creation tends to be a frequent event (in Section 4.1). When a popular news story is updated, it may be assigned a new file name. The absence of past access history for this new file makes the task of prefetching or preemptively pushing out these newly created content challenging.
- High frequency of modifications and accesses suggests prefetching or pushing the modified objects is potentially very useful. Moreover, since the popular objects stay popular for a relatively long period of time (in Section 5.2.1), it makes sense to prefetch (or push) previously popular files that have undergone modification.
- There is significance in domain membership. That is clients belonging to the same domain are more likely to share requests than clients picked at random (in Section 5.3). On the other hand, the stability of interest groups is not very high (in Section 5.2.2), so it may be very challenging to

have server push documents discriminatively to the clients. That is, if the stability of spatial locality is low, then it may be hard for server to decide where to push it if not to all clients. As part of our future work, we will investigate this issue further.

6.2.3 Implications for Web Caching

The implications of our findings for Web caching are two-fold.

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 (in 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 seems that first-time misses, though as high as 40% of the all accesses [26], are hard to cut down significantly. This is because most first-time accesses tend to be to old and unpopular documents (in Section 5.1.2), and it is very hard to accurately predict these kinds of 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 uncachable objects cachable are two major directions to improve the efficiency of Web caching, and consequently reduce the Internet traffic. Replication done under server control may be a viable solution to both issues.

7 Ongoing and Future Work

We are currently analyzing a larger content log set, investigating better heuristics for identifying client domains, and examining why the stability in interest group is low. For the longer term, we would like to study data sets from other large server sites to confirm the findings reported in this paper. We also plan to develop efficient cache consistency algorithms that are optimized based on the insights we gained from this paper.

References

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

[2] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira. Characterizing Reference Locality in the WWW. *Technical Report TR-96-11, Boston University*, 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] P. Cao, J. Zhang, and K. Beach. Active Cache: Caching Dynamic Contents on the Web. In *Proc. of Middleware'98*.

[7] 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.

[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] Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments. Anja Feldmann, Ramon Caceres, Fred Douglis, Michael Rabinovich. In *Proc. INFOCOM'99*, March 1999.

[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] D. A. Grossman, O. Frieder. Information Retrieval — Algorithms and Heuristics. *Kluwer International Series in Engineering and Computer Science*, September 1998.

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

[16] R. Jain. The Art of Computer Systems Performance Analysis. *John Wiley and Sons*, 1991.

[17] D. Li and D. R. Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol, In *Proc. of 6th IEEE International Conference on Network Protocols (ICNP'98)*. October 1998, pp. 237-245.

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

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

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

[21] 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.

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

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

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

[26] 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.

[27] G. Voelker. Personal Communication, Feb 2000.

[28] A. Wolman, G. 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.

[29] A. Wolman, G. 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

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

A Appendix

Given there are m lower-level proxies requesting the object, and x higher-level proxies, derive the average number of higher-level proxies that access the page.

Let n_j be the number of different ways a page is accessed by j Higher-level proxies, and z be the average

number of accesses the page receives from the higher level proxies. Then we have

$$z = \frac{\sum_{j=1}^{x} j * n_j}{\sum_{j=1}^{x} n_j}$$

Now let's compute n_j. As we know, n_j denotes the total number of ways of assigning m accesses from lower level proxies to j (out of m) higher level proxies. This means

$$n_j = \begin{pmatrix} x \\ j \end{pmatrix} * N$$

where N is the number of ways of putting m balls into j bins. By definition, N is the total number of ways of assigning a_i such that $a_1 + a_2 + ... + a_j = m$, where $a_i > 0$. This is equivalent to the number of ways of assigning s_i such that $s_1 = a_1$, $s_2 = a_1 + a_2$, ... , $s_j = a_1 + a_2 + ... + a_j = m$. Since $a_i > 0$, s_i is an increasing serie. Therefore $N = \begin{pmatrix} m-1 \\ j-1 \end{pmatrix}$. So we have

$$n_j = \begin{pmatrix} x \\ j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix}$$

So

$$z = \frac{\sum_{j=1}^{x} j * \begin{pmatrix} x \\ j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix}}{\sum_{j=1}^{x} \begin{pmatrix} x \\ j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix}}$$

Note that

$$\begin{aligned} & \sum_{j=1}^{x} \begin{pmatrix} x \\ j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix} \\ = & \sum_{j=1}^{x} \begin{pmatrix} x \\ x-j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix} \\ = & \begin{pmatrix} x+m-1 \\ x-1 \end{pmatrix} \end{aligned}$$

$$\sum_{j=1}^{x} j * \begin{pmatrix} x \\ j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix}$$

$$\begin{aligned} = & x * \sum_{j=1}^{x} \begin{pmatrix} x-1 \\ j-1 \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix} \\ = & x * \sum_{j=1}^{x} \begin{pmatrix} x-1 \\ x-j \end{pmatrix} * \begin{pmatrix} m-1 \\ j-1 \end{pmatrix} \\ = & x * \begin{pmatrix} x+m-2 \\ x-1 \end{pmatrix} \end{aligned}$$

With simple algebraic manipulations, we immediately get

$$z = \frac{x * m}{x + m - 1}$$