Workload Characterization and Performance Implications of Large-Scale Blog Servers

MYEONGJAE JEON, Rice University
YOUNGJAE KIM, Oak Ridge National Laboratory
JEAHO HWANG, Korea Advanced Institute of Science and Technology
JOONWON LEE and EUISEONG SEO, Sungkyunkwan University

16

With the ever-increasing popularity of Social Network Services (SNSs), an understanding of the characteristics of these services and their effects on the behavior of their host servers is critical. However, there has been a lack of research on the workload characterization of servers running SNS applications such as blog services. To fill this void, we empirically characterized real-world Web server logs collected from one of the largest South Korean blog hosting sites for 12 consecutive days. The logs consist of more than 96 million HTTP requests and 4.7TB of network traffic. Our analysis reveals the following: (i) The transfer size of nonmultimedia files and blog articles can be modeled using a truncated Pareto distribution and a log-normal distribution, respectively; (ii) user access for blog articles does not show temporal locality, but is strongly biased towards those posted with image or audio files. We additionally discuss the potential performance improvement through clustering of small files on a blog page into contiguous disk blocks, which benefits from the observed file access patterns. Trace-driven simulations show that, on average, the suggested approach achieves 60.6% better system throughput and reduces the processing time for file access by 30.8% compared to the best performance of the Ext4 filesystem.

Categories and Subject Descriptors: D.4.8 [**Operating Systems**]: Performance

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Social network services, workload characterization, measurement, modeling, filesystems

ACM Reference Format:
Jeon, M., Kim, Y., Hwang, J., Lee, J., and Seo, E. 2012. Workload characterization and performance implications of large-scale blog servers. ACM Trans. Web 6, 4, Article 16 (November 2012), 26 pages.
DOI = 10.1145/2382616.2382619 http://doi.acm.org/10.1145/2382616.2382619

This work was partly supported by the IT R&D program of MKE/KEIT [10041244, Smart TV 2.0 Software Platform], the Next-Generation Information Computing Development program through the NRF funded by the MEST (2012-0006423), and partially through the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
Authors' addresses: M. Jeon, Department of Computer Science, Rice University; Y. Kim, Oak Ridge National Laboratory; J. Hwang, Computer Science Department, Korea Advanced Institute of Science and Technology; J. Lee and E. Seo (corresponding author), College of Information and Communication Engineering, Sungkyunkwan University; email: euiseong@gmail.com.

DOI 10.1145/2382616.2382619 http://doi.acm.org/10.1145/2382616.2382619

1. INTRODUCTION

The advent of Social Network Services (SNSs) has changed the way Internet end-users participate in the Web environment. Before SNSs, users generally consumed Web content, such as text, images, videos, and audios, supplied by a limited number of producers. With the success of next-generation Web services such as blogs, wikis, video-sharing sites, and social networking platforms, however, users have become vital elements in the Internet and contributed as content suppliers by creating various types of content and sharing their activities.

As content sharing via SNSs has become extremely popular, recent years have witnessed a need to design new network and server architectures for SNSs. In the network context, reshaping of Internet traffic by SNS workloads has posed challenges for the current Internet infrastructure and content distribution systems [Krishnamurthy 2009; Rodriguez 2009]. In the server context, recent efforts have examined the negative impact of SNS workloads on server performance in terms of CPU usage [Stewart et al. 2008], cache access behavior [Nagpurkar et al. 2008], and scalability in multicore architectures [Ohara et al. 2009].

The workload characteristics of SNSs present new challenges to understanding the subsequent effects on underlying server systems. In other words, although analysis and optimization of Web servers is a well-researched area, existing techniques would not be suitable for identifying the characteristics of SNSs [Nagpurkar et al. 2008; Ohara et al. 2009]. Despite this, there has been little work for the in-depth analysis of server-side SNS workloads.

With this motivation, we have characterized real-world workloads for a large-scale blog service. The blog service represents one of the successful knowledge-sharing SNSs [Guo et al. 2009], which emphasizes on the knowledge or content sharing among blog visitors and authors (bloggers). As of February 2011, there are more than 155 million blogs in the world, with more than 1 million articles published per day. Our analysis is based on Web access logs for the blog servers hosted by *Tistory*[1], one of the most popular and large-scale blog servers in South Korea.

The blog servers in Tistory are clustered together with switches and database nodes (see Section 2.2 for more details) to serve more than 100,000 visitors every day. We observed more than 96 million HTTP requests and 4.7TB of network traffic in the logs over a 12-day period.

Using the blog server workload, we first revisit statistical findings for conventional Web workloads and provide unique results for Tistory in terms of active content creation by end-users. We then conduct a hierarchical analysis in which HTTP requests to the blog servers are viewed as aggregate traffic for the content itself and for blog articles. In particular, article analysis investigates user activities that produce and consume blog articles, which thus provides a greater understanding of processing behavior for user-oriented requests. As an example of utilizing the implication found in the article analysis, we explore the potential performance benefits from clustering of objects for a blog page in a filesystem.

Our article makes the following specific contributions.

— The study reveals many interesting findings. For example, (i) unlike conventional Web workloads [Oke and Bunt 2002; Williams et al. 2005], the file and transfer size distributions are not heavy-tailed owing to the multimedia files generated and shared by users. However, for nonmultimedia files such as HTML and image files, the distributions are heavy-tailed. (ii) User references on blog articles do not show temporal locality, which means that those read in the recent past are not likely to

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

be read in the near future. (iii) Users prefer to read articles posted with image and audio files.
— We approximated the characteristics of blog content generation and consumption using several analytical distributions. For example, (i) the transfer size of nonmultimedia files can be modeled using a truncated Pareto distribution, which indicates that large variations exist in blog content. (ii) The size of unique blog articles can be approximated using a first-order exponential distribution, which indicates that blog articles exhibit a bias towards small size.
— We investigated the workload of a large-scale blog service, including characterization of its patterns and modeling of its distributions, and compared the data with other workload characterization results. Our results provide useful information for modeling the behavior of knowledge-sharing SNS workloads, developing workload generators, and designing new systems to run knowledge-sharing SNSs.
— We studied filesystem optimization to improve processing time for file requests and system throughput of blog servers. Specifically, from an observation that files in an article have strong access locality, we explore possible performance optimization by colocating article files for the same blog page into contiguous disk blocks. Simulation shows that this optimization outperforms a variety of scenarios in Ext4.

The remainder of the article is organized as follows. Section 2 describes the background to blog services and the server architecture. Section 3 presents overall features observed for the workload. Sections 4 and 5 explain file-level and article-level characterization, respectively. Section 6 discusses filesystem optimization. Section 7 describes related work, and we conclude in Section 8.

2. BACKGROUND

2.1. Overview of Blog Web Pages

The main content in a blog page is a series of articles. The articles in a blog are typically arranged in reverse chronological order with the most recent article appearing at the top of a blog. Due to this inherent organization, the information on new content is easily accessible to potential visitors.

There are several ways that make blog users reach new blog content.

— *Direct Visits*. A user may access a blog by typing its URL in a Web browser and loads the blog contents into the browser.
— *Web Feeds*. New contents of a blog are published via Web feeds, and later they are accessed by the subscribers of the blog's Rich Site Summary (RSS) service through RSS feed readers.
— *Blog Search Engines*. Blog search engines identify blogs, index content, and enable users to search for blogs, their authors, and the relationships among them.
— *Trackbacks*. A trackback is an anchor to an article that can be embedded in or attached to other articles. A user can access an article by following the trackback of the article from other related articles by following the embedded or attached trackback link.

All four methods described before not only facilitate the grouping of blog community, but also ultimately accelerate the interactions between content producers and consumers.

In general, a *blog page* shows its contents with a logically uniform structure, as illustrated in Figure 1. A *template* is used to form the design of a blog page using a combination of images, CSS, and JavaScripts. An *article* is the unit of a blog posting and consists of an *article text* and *article files*; the article text is a text message

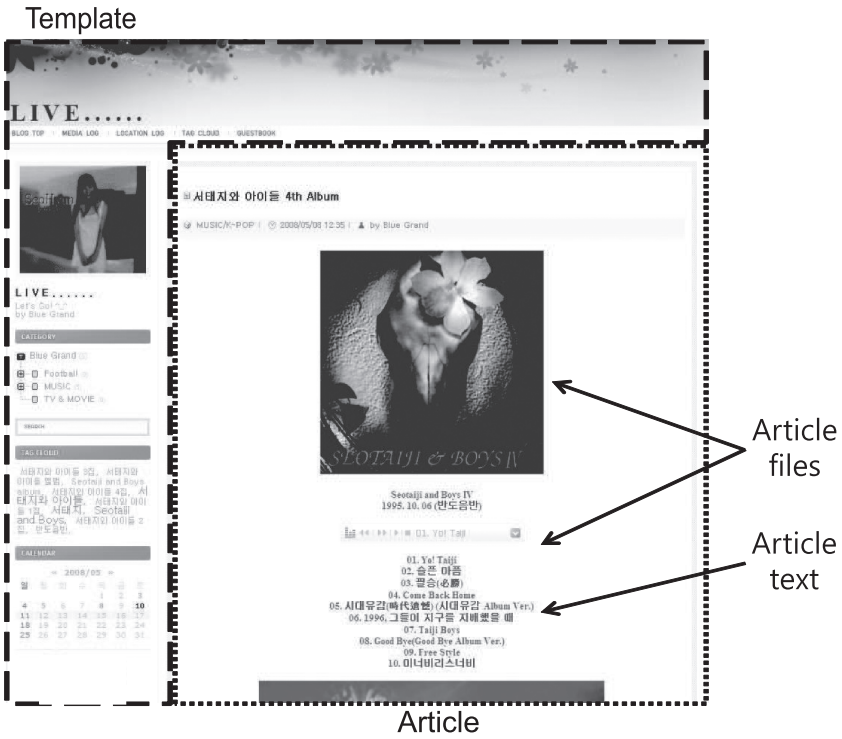


Fig. 1. An example of a blog page.

written by the author of the blog, whereas the article files refer to the files uploaded and incorporated with the article text by the author. The article files can additionally be categorized into *article image*, *article audio*, and *article video*, according to the file type. All blogs that use the same blog template show the identical backgrounds on all their blog pages. Also, if a user visits several blog pages in a certain blog, only the article is changed because all blog pages that commonly belong to a certain blog share the same template.

Blog servers provide users with blog contents that are produced in two manners; one is *dynamic* and the other is *static*. While the *dynamic content* is generated on-the-fly by server-side scripting when it is sent to the users, the *static content* is stored in a disk and fetched using filesystem operations. In Tistory's case, rendering main (or front) blog pages requires server-side scripting to dynamically embed objects such as the article text, comments, and replies, which are dynamic contents and all stored in the server database. All other contents in the blog page that are otherwise stored in the filesystem belong to the category of static contents.

A blog page includes only one article text in most cases. However, if a blogger changes the blog's parameters using configuration tools, access of a particular blog page may retrieve several articles conglomerated in that page. This poses a challenge in article analysis because all articles in a blog page must be identified. The details of the methodology to address this challenge are explained in Section 5.

2.2. Server Architecture and Data Collection

The system of Tistory consists of a set of switches, Web servers, and database nodes as shown in Figure 2. A switch node controls the incoming requests and forwards

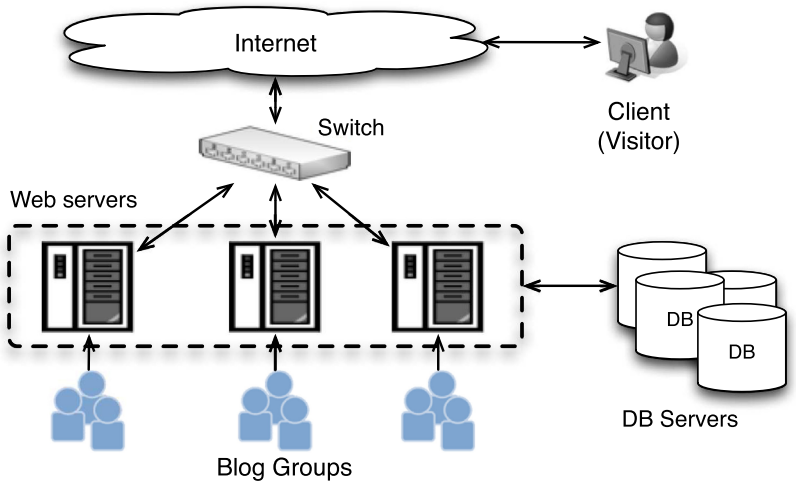


Fig. 2. The architecture of blog service systems.

Table I. Description of Fields in a Log Entry (Every log entry follows the Extended Common Log Format (ECLF) format)

Field	Description
VirtualHost	The URL assigned to a particular blog
ClientHost	The IP address of the client who issued the request
Date&Time	A time stamp recorded at the request reception time
Request	The actual HTTP request indicating the requested file and method to process the file
StatusCode	A HTTP response code signifying the result of processing the HTTP request
Size	The length of the transferred file
ProcessingTime	Time taken to process the request

them to the Web servers depending on their destinations. Each Web server manages a group of blogs and retrieves both the static and dynamic contents corresponding to the received requests. Database nodes, being connected to Web servers, store all textual data necessary for composing dynamic contents. Web servers typically submit queries to the database nodes to retrieve necessary textual data. The reverse proxies are not installed in our server architecture. To the best of our knowledge, this is a typical blog server platform.

The raw material for analysis is the access logs collected from all the Web servers for 12 consecutive days. The log entry is in ECLF (Extended Common Log Format) format and has several fields as presented in Table I. The server is configured to have a time granularity of one second.

After eliminating those requests from the access logs in which the server did not return the file to the client, total 50,118,065 requests were obtained, wherein we witnessed the existence of 948,290 visitors and 11,629 blogs over the observed time interval. The reduced log data maintains all system-wide events occurring to both the template and the article, particularly the content transfers between clients and Web servers.

Table II. Breakdown of HTTP Methods in the Workload

Method	Requests (#)	Proportion
GET	50,118,065	97.35%
POST	1,314,064	2.55%
Others	52,932	0.10%

Table III. Statistics for File Types

	html	Java script	CSS	Image	Audio	Video	Flash
Files (#)	123	1,438	7,989	401,921	15,414	409	1,208
Average size (KB)	18.42	3.042	7.020	132.8	4,427	4,163	603.2
Std. dev.	30.17	5.067	7.588	269.6	2,164	3,645	1,301

3. DYNAMICS OF BLOG POSTING AND BROWSING

This section presents a statistical analysis of user activities and the constitution of blog data to gain an understanding of the overall characteristics of a blog workload in comparison to traditional Web services and Youtube, the world's largest video-sharing site.

3.1. User Activity Analysis

One way of observing user activities is to analyze HTTP methods in the *Request* field in log entries. Table II shows the breakdown of HTTP methods found in our workload. The `GET` request is used to retrieve files from the Web server. The `POST` request is used when end-users intend to upload Web contents, make trackbacks, or post comments.

Table II shows that `GET` accounts for the majority of the total requests (97.35%), whereas `POST` accounts for only 2.55% of occupancy rate. The most common use of `POST` is for uploading content such as articles, images, and audio and video elements (348,530 requests). Posting comments and enabling trackbacks follow uploading content with 317,211 and 187,067 trials, respectively. Although the 2.55% for `POST` seems insignificant, it clearly reveals the tendency of blog users to actively participate in publishing content and online interactions in comparison to traditional Web services. For instance, in the case of Youtube traffic, only 0.12% of requests were `POST` [Gill et al. 2007], an order of magnitude less than that observed for `POST` requests for Tistory blogs. In the workload for the 1998 World Cup Web site, only 0.06% of requests were `POST`.[2]

3.2. File Type Analysis

Since users actively participate in blogging, file types and their place on the server are greatly affected by what content blog authors prefer to share. Content elements are typically accessed by a large number of independent visitors. Therefore, access patterns to these elements are unpredictable compared to traditional Web workloads. As a result, use of server resources, such as disks and networks, also changes according to access patterns.

Table III shows various types of content stored on Web servers, and Table IV shows their influence on the use of network or storage bandwidth. Cumulative Distribution Functions (CDFs) of file size are shown for the file types in Figure 3.

Two multimedia file types, video and audio, show almost the same average file size of 4.1–4.4MB and a skewness in the range 3–5MB. Image files dominate the number

[2]The log data are available at http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

Table IV. Number of Requests and Bandwidth Usage for Each File Type ("Others" represents HTML, CSS, video and flash files, which together account for less than 1%. "Dynamic" denotes dynamic files)

	gif png	jpg bmp	audio	Java Script	Dynamic	Others
Request (%)	44.3	17.0	1.5	5.9	28.2	3.1
Bandwidth(%)	3.2	24.2	61.0	0.8	5.9	4.9

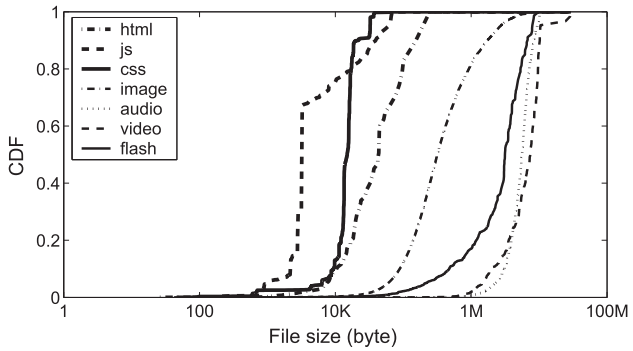


Fig. 3. CDFs of file size by file type.

of requests. It is likely that many bloggers upload audio rather than video content to their blogs because videos are often larger than the blog upload limit, which is 10MB per file. Instead, authors tend to embed content from other video streaming services such as YouTube in their blog articles.

The average size of transferred files is approximately 100KB (Table V), which is much greater than results reported for earlier Web workloads [Bent et al. 2004; Williams et al. 2005]. For example, Williams et al. reported that most transferred content files fell in the range 1.3–3.7KB [Williams et al. 2005]. The situation is not that different in current Web-based service workloads. In a most recent study, the average transfer size was estimated to be 19KB in the workload of a large enterprise and 3KB in the workload of a university [Gill et al. 2011].

As shown in Table IV, file transfer shows three additional properties that differ from previous studies [Bent et al. 2004; Faber et al. 2006; Oke and Bunt 2002; Williams et al. 2005].

— Transfer of audio elements accounts for more than half of the network bandwidth usage, although the number of requests for audio elements is relatively small.
— The network bandwidth consumption for image transfer is much smaller than for conventional Web workloads, even though the percentage of image file requests is comparable.
— HTML requests are drastically reduced from 10–30% to 0.003% owing to the dominance of dynamic Web pages, which notably replace static Web pages.

Even with the reduced bandwidth use for image files, photographic images such as jpg and bmp are still responsible for a quarter of the total usage. Specifically, they account for 17% of total requests and 24.2% of total bandwidth usage, and thus rank second in terms of bandwidth usage. The influence of videos is still minor because large video files in blog articles are stored on and streamed from other video streaming services, as noted earlier.

Table V. Comparison of Static and Dynamic Content Elements in Terms of Processing Time and Transfer Size

	Total	Static	Dynamic
Requests (#)	50,118,065	36,138,738	13,979,327
Total processing time (s)	26,998,466	22,573,520	4,424,946
Average processing time (us)	538,697	624,635	316,535
Std. dev.	138,475	10,738	557
Total transferred data (GB)	4729.53	4452.53	276
Average transfer size per request (KB)	98.95	128.74	20.97
Std. dev.	655.49	739.87	336.12

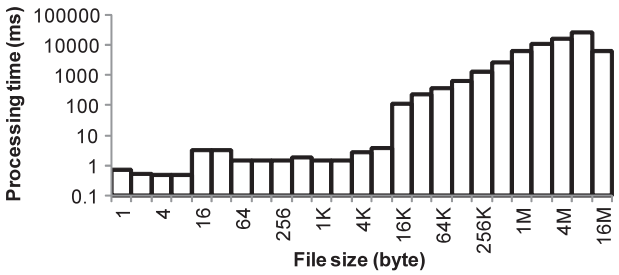


Fig. 4. Request processing time for static files according to size. The horizontal axis is on a log scale.

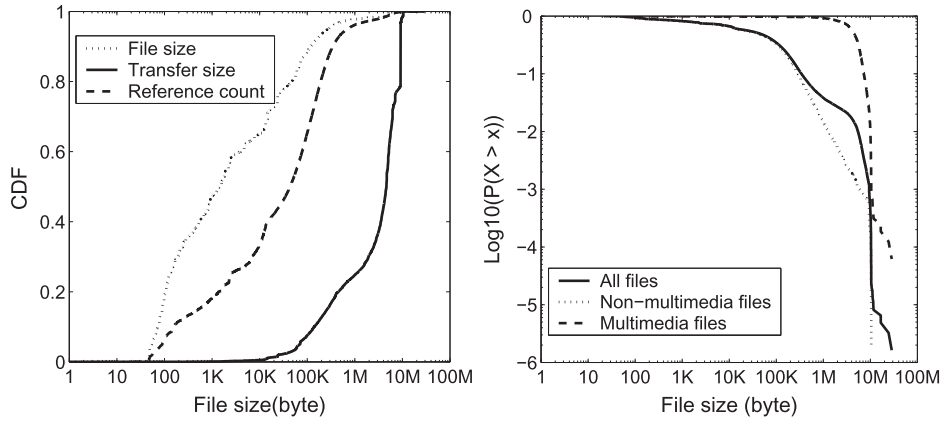
3.3. Static Versus Dynamic Content

Processing of requests for dynamic content often involves a substantial amount of processing time to execute server-side scripts. Therefore, for Web servers that serve dynamic content: (1) processor performance, rather than network or disk performance, often determines the overall system throughput, and (2) the time taken to process dynamic content requests is at least an order of magnitude greater than for static content requests [Challenger 1996; Dingle et al. 1999; Holmedahl et al. 1998]. Our results reveal that this characteristic does not exist in the blog workload where requests for dynamic content constitute 28% of the total number of requests (Table V).

In this section, we compare the processing time and bandwidth usage for each content type.

Our results show that static content, which dominates network usage, also takes a longer processing time than dynamic content. Approximately 94% of the total bandwidth (>4.4TB) is used to deliver static content. The average processing time is 0.31 and 0.62 seconds for dynamic and static content, respectively. This difference shows that execution of a server-side script usually takes a shorter time than file fetching and delivery. Moreover, the total time required to deal with all static content requests constitutes 82% of the total processing time for all requests. The main reason for the faster processing of dynamic content may be smaller size of the content (along with improvements to computing technology, e.g., processor, memory). Thus, the blog workload differs from traditional workloads, for which the presence of dynamic content significantly slows down Web server performance [Holmedahl et al. 1998; Iyengar and Challenger 1997].

The time required to deal with static content requests depends on the file size. We investigated the relationship between file size and processing time. As shown in Figure 4, the time to process static files of <16KB is less than 10 ms on average, but increases proportionally with content size. Processing of a request for static content of 256KB takes up to 1.3 seconds.



(a) CDF (b) LLCD

Fig. 5. (a) CDFs for static files with respect to file size, reference counts, and transfer size; (b) LLCDs for static content with respect to file size for multimedia files, nonmultimedia files, and both together.

So far we have identified some fundamental differences between blog and conventional Web service workloads. Hereafter, we focus on static content for further analysis, since we identified this as the major contributor to server utilization. In addition, some recent research results have revealed that processing of static content should be well-designed in terms of system performance optimization because processing of this type of content may greatly increase stall cycles owing to both huge data cache misses [Nagpurkar et al. 2008] and large network flows within servers [Veres and Ionescu 2009].

4. FILE-LEVEL ANALYSIS OF STATIC CONTENT

In this section we describe the characteristics of static content at file level. File-level analysis has been widely applied in research on Web traffic characterization [Arlitt and Jin 2000; Arlitt and Williamson 1997; Faber et al. 2006; Oke and Bunt 2002] and previous studies were referred to in designing our analysis approach.

We first examined file and transfer size distributions to assess the use of storage space and network bandwidth, respectively. We then analyzed the frequency of references to individual static files to identify nonuniform file referencing phenomena.

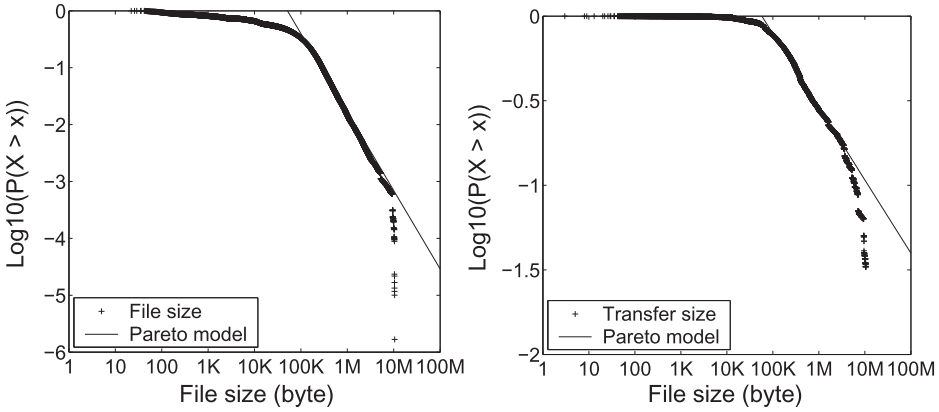
We use the terms "static content" and "static files" interchangeably. Hereafter, multimedia content refers to audio and video files, whereas nonmultimedia content denotes image, Java script, CSS, and Flash application files.

4.1. File Size and File Transfer Size

Figure 5(a) presents Cumulative Distribution Functions (CDFs) for static content with respect to file size, reference count, and transfer size. File transfer size denotes aggregate bandwidth usage for a specific file size, which was obtained from the simple calculation $file\ size \times reference\ count$.

Figure 5(a) shows that a small number of large files require high server bandwidth, whereas a large number of small files require low server bandwidth. For instance, 92% of total static files are smaller than 400KB, whereas their bandwidth demands account for less than 20% of the total server bandwidth.

As expected, large files posted by users significantly contribute to the bandwidth usage; in particular, among files larger than 400KB, photographic images (59%) and audio files (33%) predominate. Moreover, Figure 5(a) shows that file size visually follows



(a) File size (b) File transfer size

Fig. 6. LLCD transformation for nonmultimedia files.

Table VI. Approximation Results for File Size and Transfer Size Distributions for Nonmultimedia Files

	File transfer size	File size
Tail index(α)	0.43	1.37
Goodness of fit(R^2)	0.98	0.99

a heavy-tailed distribution, whereas file transfer size does not show this trend. We confirmed this observation by modeling the data using a Pareto distribution (see the Appendix), which is a well-known model for representing heavy-tailed distributions. Thus, we transformed CDF plots for file size and transfer size to Log-Log Complementary Distribution (LLCD) plots [Crovella and Bestavros 1997] and investigated their R^2 *goodness-of-fit* values.

The results revealed that LLCDs of file size and transfer size could not be modeled using a Pareto distribution. Surprisingly, this observation is in contrast to conventional Web workloads, for which file size and the transfer size show heavy-tailed distributions [Arlitt and Williamson 1997; Oke and Bunt 2002; Williams et al. 2005]. In further analysis, we modeled the LLCDs for file size and transfer size for multimedia and nonmultimedia files. We found that only nonmultimedia files showed heavy-tailed distributions for file size and transfer size.

Figure 6 shows file size and transfer size LLCDs for nonmultimedia files. It is evident that both parameters can be neatly modeled using a truncated Pareto distribution [Aban et al. 2006] with a natural upper bound that curtails the tail.

The tail index α with confidence level R^2 is shown in Table VI. The LLCD tail length is threefold greater for file size than for transfer size.[3] Thus, although the distributions closely fit the Pareto distribution model with high R^2 values, the α values differ. The values indicate that the distribution is much more heavy-tailed for transfer size than for file size. It is obvious that this result can mainly be attributed to the dominant bandwidth usage for photographic images (59% of the total).

Figure 7 and Table VII show LLCDs of file size and transfer size and approximation parameters, respectively, for multimedia files for completeness. As evident in the figure and the table, multimedia files are not modeled using a Pareto distribution.

[3]The blog service has a file size restriction of 10 MB. This might affect the tail length.

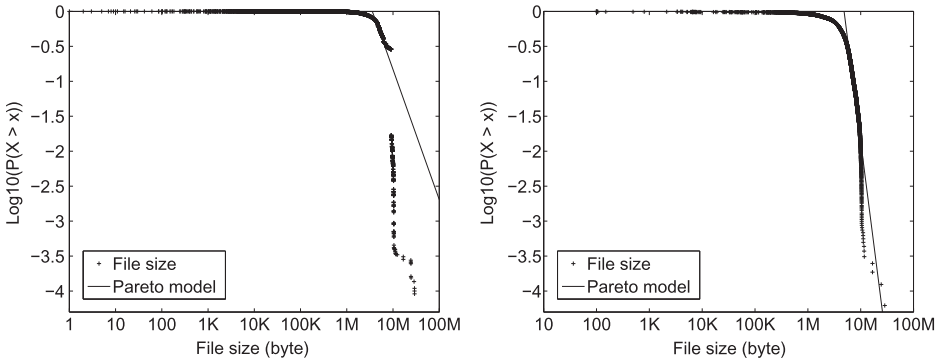


Fig. 7. LLCD transformation for multimedia files.

Table VII. Approximation Results for File Size and Transfer Size Distributions for Multimedia Files

	File transfer size	File size
Tail index(α)	1.8	5.9
Goodness of fit(R^2)	0.61	0.93

Specifically, they reveal either a poor R^2 goodness-of-fit (as shown in transfer size) or an inappropriate tail index value (as shown in file size).

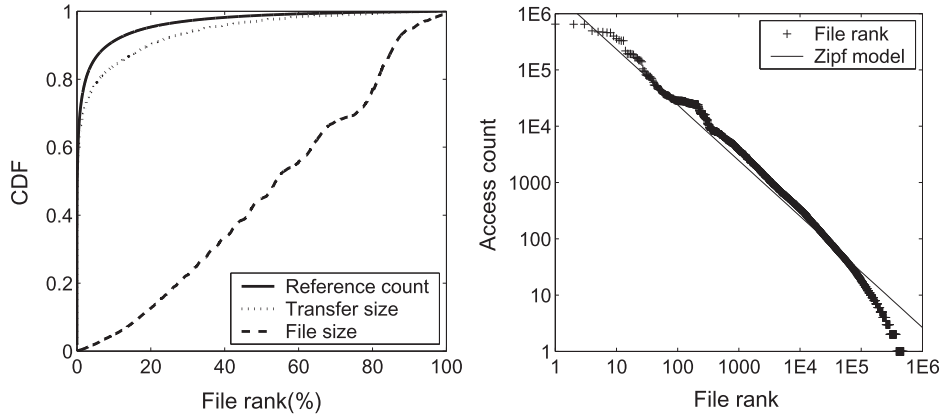
Our results imply that the characteristics of conventional Web servers may not be perfectly suited for blog service workloads. Several studies have revealed that the file size and transfer size of Web workloads are heavy-tailed. However, our analysis of a large-scale blog workload shows that both file size and transfer size distributions are heavy-tailed only for nonmultimedia content. In effect, blogs may generate significantly different request patterns for the underlying server systems.

4.2. File References

To investigate file reference characteristics, we analyzed the relationships between some popular files and actual data for storage and network bandwidth usage. Files were sorted according to reference count and we assumed that the most popular file occupied the highest rank. Disk space usage, network usage, and reference counts were then combined in a CDF plot (Figure 8(a)).

It is evident that 10% of the most frequently referenced files account for 84% of the data transferred over the network and 91% of the total requests. However, these files account for only 4.8% of the overall storage allocation.

An immediate implication of this result is the effectiveness of caching, since storage of only a small proportion of popular files would lead to a significantly higher successful hit ratio. By storing only 10% of long-term popular files that occupy up to 6.1GB of storage space, a cache system could save up to 91% of request service and 84% of traffic volume incurred by static content requests in the ideal case. Furthermore, a surprisingly high proportion of resources were used for photographic and audio files, amounting to 6.0GB of storage space and 3,648GB of traffic volume. We observe that both large files (>400KB) and popular files (top 10%) influenced a large proportion of traffic volume. This is because 70% of the total bandwidth is used for files that lie on the intersection between the two groups.



(a) CDF (b) Zipf analysis

Fig. 8. Reference count according to file rank. (a) CDF of file size, reference count, and transfer size; (b) Zipf analysis (access count versus rank).

A typical method used to approximate popularity distributions is Zipf's law [Zipf 1949], according to which the number of file references consistently decreases with the file rank. If (R) represents the file rank, then the number of references is $P \backsim R^{-\beta}$, where β is the slope value.

The formula shows that the number of references is inversely proportional to the file rank. In other words, if the popularity follows Zipf's law, a graph of log-scaled R versus log-scaled P will be linear. Figure 8(b) shows that Zipf's law provides a good approximation our file popularity, with β and R^2 close to 0.99 and 0.98, respectively.

5. ARTICLE-LEVEL ANALYSIS OF STATIC CONTENT

This section explains the characteristics of static content at article level. We first describe the methodology used to extract article access logs from the file access logs used in the previous section. Using these article access logs, we investigate the distributions for uniquely identifiable articles to determine user behavioral characteristics when producing and consuming articles. We also explore the access channels to an article to gain an understanding of how users read articles.

5.1. Methodology Overview

We collected *article access logs* in two phases. The first phase takes the blog page organization into account. When visiting a blog, a client usually reads one article and then, after a while, another article. This sequence shows a unique request pattern, as shown in Figure 9. For a clear explanation, we define two terms: (1) *active interval*, the time between two consecutive requests for files within an article; and (2) *inactive interval*, the time between two consecutive requests for files in different articles. We measured average active and average inactive intervals for requests for several blogs. The difference between the interval types is marked, so we can easily distinguish them; active intervals are typically milliseconds and inactive intervals are typically tens of seconds. Using these two intervals, we grouped data from the raw access logs into article access groups.

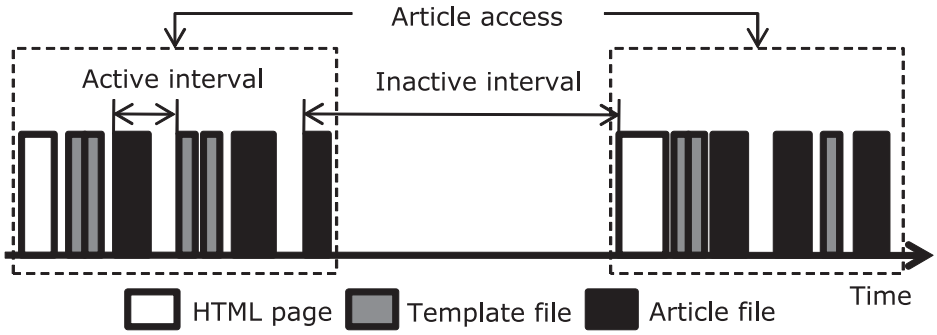


Fig. 9. Example of an access pattern when a client reads several articles in a blog.

The following points explain the detailed procedure for extracting the *article access groups*.

Step (1). Select a collection of file access logs corresponding to a unique pair of *VirtualHost* and *ClientHost* fields in the log entries.

Step (2). Gather requests for a blog page and all susbequent embedded file requests that appear until the next request for another main blog page arrives or a file request is submitted over a time threshold of 5 seconds. The time threshold is chosen based on analyzing the distribution of intervals and finding a knee in the behavior.

Step (3). Given the requests for a blog page from step (2), group them as an article access group. Then return to step (2) to pick up requests for another article access group. Repeat this until all the file access logs are processed.

Some article access groups acquired from the first phase included references to several articles. For instance, the front page of a blog sometimes contains more than one article to encourage visitors to read several recent articles. It is not possible to identify how many and which articles are on a page from the information in the raw access logs. Thus, we used a second phase to correct these article access groups.

The second phase uses a *file matching* method, as shown in Figure 10. Suppose we have three article access groups, A, B, and C, each containing article files identified by their name. By comparing the article file names in A and B, we can determine that A includes B. In the same manner, we can decide that C is also included in A. Finally, A is divided into two article access logs, B and C. Using this method, we refined the article access groups from the first phase and obtained the article access logs.

After eliminating a few articles (<3%) of unknown size, we obtained 87,332 unique articles stored in the blog servers and 1,088,210 accesses to these articles during the study period. Table VIII summarizes the article data, from which we can estimate that each article was read approximately 12.5 times on average.

5.2. Characteristics of Blog Articles

We analyzed the inter-reference time for article accesses to investigate whether temporal locality exists for subsequent accesses.

Unique Article. Figure 11(a) shows a histogram of the number of article files for unique articles. An article contains 2.74 files on average, with median and standard deviation of 1 and 8.89, respectively.

The number of articles containing fewer than four files accounts for 87% of the total, which indicates that bloggers prefer to publish articles with a small number of

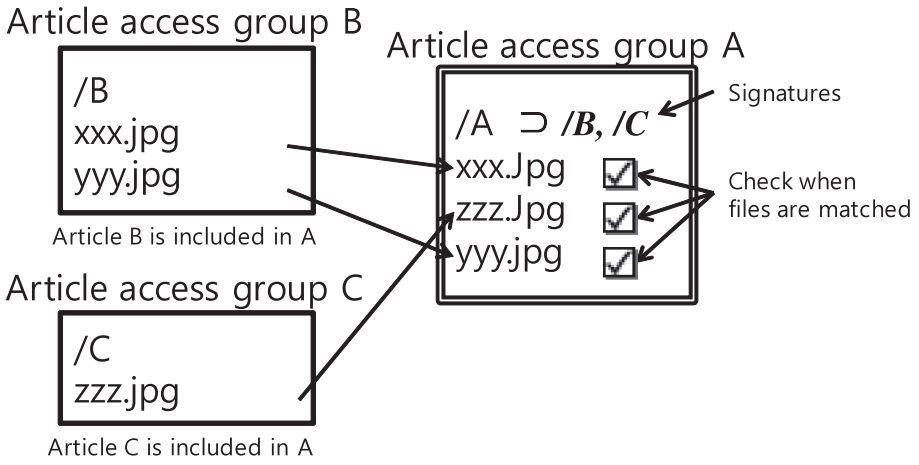


Fig. 10. File matching method to deal with cases in which article access group A contains access counts to article access groups B and C.

Table VIII. Summary of the Article Statistics

	Articles (#)	Article files (#)
Unique articles	87,332	235,586
Access counts for unique articles	1,088,210	2,909,894

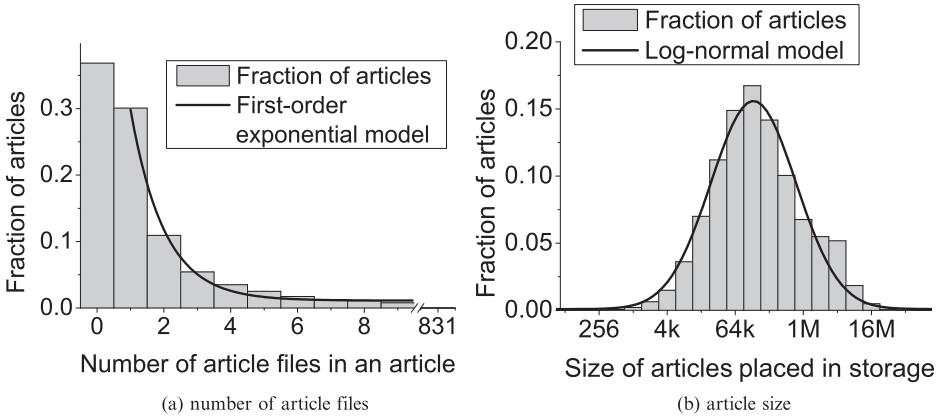
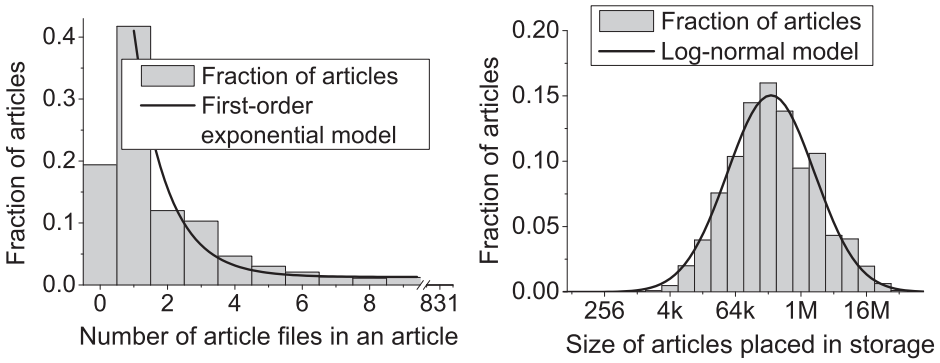


Fig. 11. Histograms of (a) number of article files and (b) article size for unique articles.

files. We can approximate the histogram in Figure 11(a) with a first-order exponential distribution ($0.01 + 0.77e^{-x/1.0}$) except for zero-file articles (i.e., articles with only text).

The goodness-of-fit (R^2) value of 0.99 confirms that the data show strong correspondence to the distribution. This approximation explains that the number of articles in blogs exponentially decreases as the number of article files increases.

Figure 11(b) shows a histogram of article size for unique articles. The average size is 1.1MB, with median and standard deviation of 203KB and 4.6, respectively. Article size mostly falls in the range from 32KB to 2MB (74% of the total files).



(a) number of article files (b) article size

Fig. 12. Histograms of (a) number of article files and (b) article size for article access counts.

We can approximate the histogram with a log-normal distribution [Limpert et al. 2001] ($\mu = 12.3, \sigma = 1.80$, natural log), as shown in Figure 11(b). The R^2 value is 0.98, which confirms that the distribution shows strong correspondence to the data.

Access Frequency for Unique Articles. Figure 12(a) shows a histogram of the article access frequencies. The average frequency is 2.67, with a median of 1 and standard deviation of 6.12.

Articles composed of fewer than four files account for 86% of the total article accesses. Comparison of the percentage access for zero-file and other articles reveals that blog visitors prefer to read articles posted with article files such as image and audio elements.

The data in Figure 12(a) can be approximated using a first-order exponential distribution ($0.01 + 1.10e^{-x/0.98}$) except for accesses to zero-file articles. The R^2 value of 0.97 confirms there is a strong fit to the distribution.

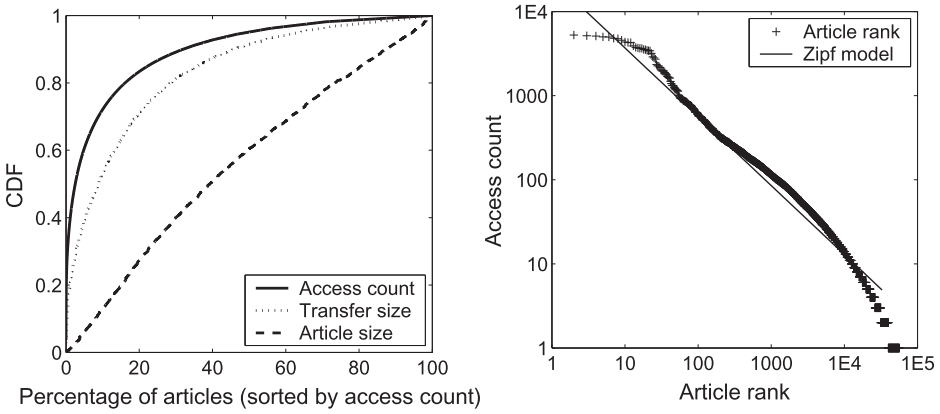
Figure 12(b) shows a histogram of the size of articles. The average size of articles is 1.07MB, with a median of 202KB and standard deviation of 3.46. Articles in the size range from 32KB to 2MB account for approximately 74% of total file accesses.

We can approximate the histogram with a log-normal distribution ($\mu = 12.3, \sigma = 1.80$, natural log), as shown in Figure 12(b), with a strong R^2 value of 0.98.

Similar to the exponential distribution, the log-normal distribution is common when the average is low and the variance is large. Our analysis for blog articles consistently shows that article access counts and creation patterns for most users are highly concentrated on small articles or articles with a small number of files.

Article Reference Behavior. We conducted a rank-based analysis as in Section 4.2. Articles were sorted according to access count, with the most popular one given the highest rank. Figure 13(a) shows CDFs for storage, network bandwidth usage, and access frequency for article files.

It is evident that the 10% of articles with the most frequent access are responsible for 73% of total reference counts and 53% of the bytes used to transfer total articles. These results are much lower than those for file popularity, with 90% of requests and 84% of bytes attributed to the top 10% most frequently requested files. Owing to the diluted concentration on the articles, popularity modeling using Zipf's law yields a lower slope for article popularity than for file popularity. The β value is 0.82 with, an R^2 value of 0.98 (Figure 13(b)).



(a) CDF of article size, access counts and transfer size (b) Zipf analysis (access count versus rank)

Fig. 13. Reference count according to article rank.

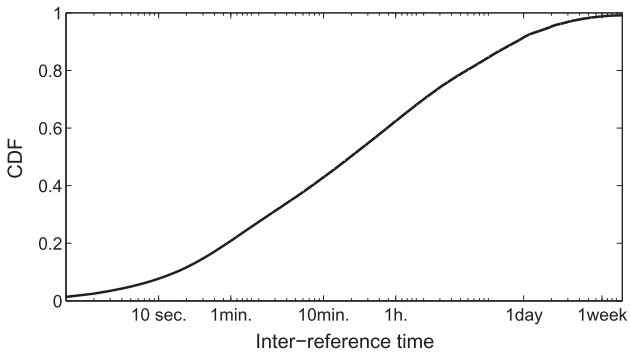


Fig. 14. CDF of article inter-reference time.

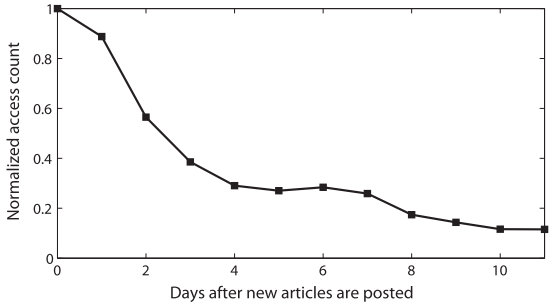
Temporal Locality. We found that article access patterns exhibit weak temporal locality. Figure 14 shows a CDF of inter-reference time for articles. Articles that are revisited within 30 minutes and 1 hour account for 31% and 62% of the total, respectively.

A notable trend in the graph is a monotonous increase as the interval increases, and thus no rereferencing hot spot is evident. We might expect that temporal locality would occur for article accesses because the top 10% popular articles account for 73% of the total reference counts. We speculate that this delayed locality may be strongly affected by active content creation, whereby article references are distributed among various articles.

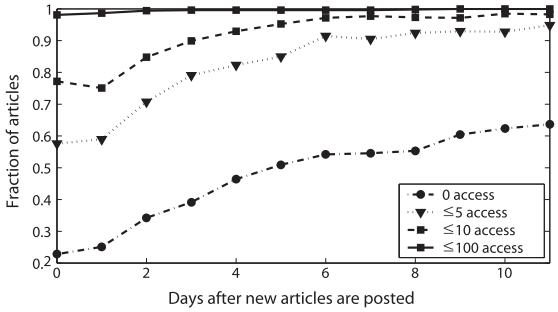
5.3. Evolution of Article Access Patterns

We continue our discussion on article access patterns by focusing on how the popularity of newly posted articles evolves over time and how fast or slowly this changes. For this analysis, we used articles published on the first day of the study period.

5.3.1. Evolution for New Articles. We first analyzed how access counts for new articles changed over a period of 12 days. Figure 15 shows the total access counts of the new articles, which is generated on the first day, with values normalized to the counts observed on the first day. The access counts decreased sharply to 35.8% during the first



(a) Change in total access counts over time for new articles. The access counts are normalized to those observed during the first day of article posting.



(b) Change in the proportion of articles accessed according to the number of accesses ("≤ 100" denotes articles accessed 100 or fewer times on a particular day).

Fig. 15. Popularity evolution over a period of 12 days for newly posted articles.

3 days, but decreased gradually thereafter. This evolution suggests that the popularity of newly published articles is ephemeral.

Newly posted articles do not have a uniform popularity in nature. Moreover, the access counts for individual articles change over time. To investigate this, we classify the articles into different range of access count and examine how the occupancy of each range evolves over time.

Figure 15(b) shows plots of the proportion of new articles accessed ≤ 100, ≤ 10, ≤ 5, and 0 times per day over a 12-day period. A notable trend in the plots is a consistent decrease of the popularity over time. For example, on the first day, 58% and 78% of the articles were viewed 5 and 10 times or less, respectively, meaning that 42% and 23% of the articles had access counts more than 5 and 10. After 11 days, there were only 5% and 2% of the articles that had access counts more than 5 and 10. Furthermore, as much as 64% of the new articles were never accessed at the last day of the study period. In summary, the majority of new articles do not draw strong interest from blog readers over a long period of time.

Some new articles seem to be popular as soon as they are posted and gather visitors steadily over time. To gain a better understanding of article popularity, in the next section we discuss the evolution of access patterns with a focus on changes in reference rates and inter-reference times. As a sample set, we selected articles that were viewed at least 100 times during the first 2 days.

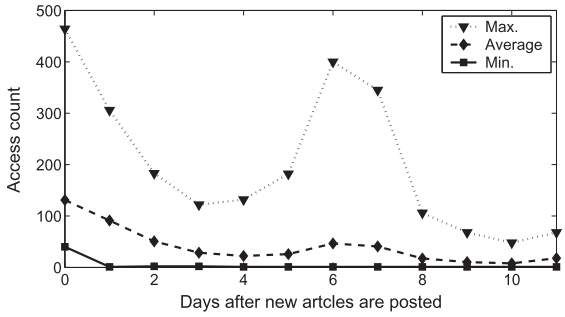


Fig. 16. Changes in maximum, minimum, and average access counts for popular articles over 12 days.

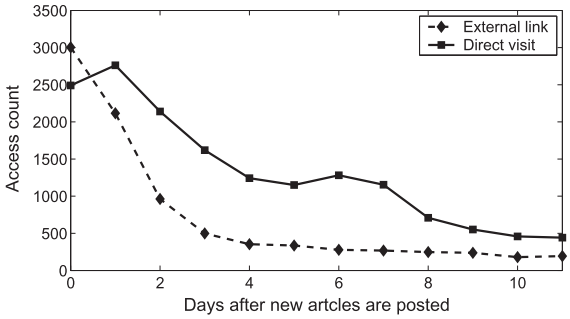


Fig. 17. Comparison of access counts for external links and direct visits.

5.3.2. Evolution for Popular Articles. We analyzed changes in the maximum, minimum, and average access counts over time. Figure 16 shows that on the first day, the maximum and average counts were 460 and 103, respectively. However, the average access count decreased over time and reached a constant level of $<$50 after 2 days. This shows that even if an article received a high number of views during the first 2 days after posting, initial popularity does not guarantee future popularity.

The gap between the maximum and average plots indicates that only a very small number of new articles attracted visitors, with the remainder ignored. The maximum access count was consistently greater than 50 for 12 days, but suddenly peaked between 6 and 7 days after publishing. These results reveal that the maximum access count is mainly influenced by a particular article in which the blogger posts a series of interesting articles that draw readers back on a weekly basis. Identification and favoring of such bloggers would be of benefit in managing a blog service.

5.3.3. Effects of the Incoming Path. An understanding of what directs visitors to a new article and the speed of their effect on access counts provides useful information on the influence of a blog article. We classified the ways of accessing an article into two categories: *external links*, whereby a user visits a blog through online interactions such as RSS, trackbacks, and recommendation sites; and *direct visits*, whereby a user types the blog address into a Web browser to view articles directly.

The use of external links quickly decreased over time, whereas the use of direct visits decreased at a slightly lower rate, as shown in Figure 17. On the first day, the count for access via external links was approximately 1,000 greater than for direct visits, but this difference fell steeply to 499 after 3 days. This indicates that once an article is posted, the information propagates quickly via external links. By contrast,

the number of new articles read via direct visits decreased from 2,762 on Day 1 to 1,150 on Day 5. In summary, access to new articles through external links, which are devised to promote user interactions, is more ephemeral than direct access to articles.

6. DISCUSSION

We now describe an example of performance improvement for a blog server filesystem based on the implications obtained from our analysis. Our approach stems from the question of how the strong temporal locality of file accesses for an article can be utilized to optimize filesystem performance. The scheme discussed in this section is colocation of those files on contiguous disk space to maximize the benefits of the locality. We describe expected benefits of the scheme in terms of improved throughput and processing time for file requests through simulations.

6.1. Filesystem Optimization for Blog Servers

Slow disk I/Os have been a major performance bottleneck in large-scale Web servers [Barford and Crovella 1999; Kant and Won 1999]. The performance gap between the CPU and disk, which is already significant, is growing because of the widespread use of multicore processors. The gap is also increasing as applications become more data-intensive.

One solution to overcome this problem is the provision of better in-memory buffer cache performance. However, the performance benefit is limited because the fraction of I/O requests absorbed by the cache is far less than the proportional increase for larger cache size [Wachs et al. 2007].

For our blog workload, user accesses to unique articles exhibit a strong tendency whereby *all of the files (or otherwise none of the files) for an article are transmitted from the blog servers to the clients within a short time frame*. We analyzed how many articles supplied by a server are not included in this article transfer pattern. For transmission of articles with more than one article file, partial file delivery was observed for only 3.2%. All of the files for an article or none of them were transferred in 73.2% and 23.5% of cases, respectively. This implies that an article, rather than a file, can be thought of as the unit to be fetched from the disk.

An intuitive approach that uses this observation is to prefetch all files associated with an article when any of those files is initially requested, in anticipation that all other files for the article will be subsequently requested. This prefetching scheme can be realized using hints given by applications [Patterson et al. 1995; Tomkins et al. 1997] or repeatable access patterns captured in applications [Li et al. 2004].

However, a simple prefetching scheme would not be significantly more effective because: (1) sequentiality at file abstraction level may not translate to sequentiality on disk as the filesystem ages, and 2) multiple outstanding file requests that independent clients simultaneously generate will be interleaved on the disk, which makes the disk access patterns more random. These issues are expected to be worse in a workload predominated by large files.

Cluster-based filesystems were introduced to optimize I/O performance by grouping files with strong temporal locality into a cluster, which is stored to or fetched from a contiguous disk region [Shriver et al. 2001; Wang and Li 2003]. This mechanism retains the benefits of prefetching but effectively overcomes the nonsequentiality and interleaved access issues discussed earlier.

Cluster-based systems are expected to be very effective for blog server filesystems because all files for an article exhibit the access locality. Therefore, such files, if composed into a cluster and prefetched together when any one of the files is initially

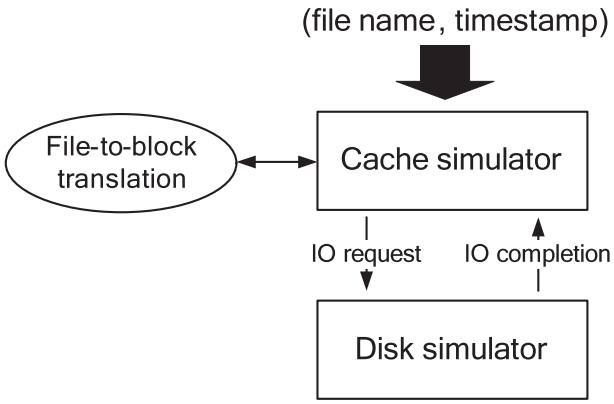


Fig. 18. The simulator architecture.

requested, will be retrieved by clients within an adequate time frame without performance degradation due to mispredictions.

6.2. Evaluation

We discuss the overall impact that the proposed optimization would have on the blog workload. A simulator was built to evaluate the effects of the proposed cluster-based scheme on the server performance. The simulator is largely composed of a buffer cache manager and a disk simulator as shown in Figure 18. Data access miss in the buffer cache should be requested to the disk simulator. We replayed our traces of all file access requests on top of the buffer cache manager. Specifically, each file access was first transformed into an access (or multiple accesses) to data blocks used by the file and then issued based on the timestamp logged in the traces. We used debugfs program to find each file's block information under a certain filesystem layout.

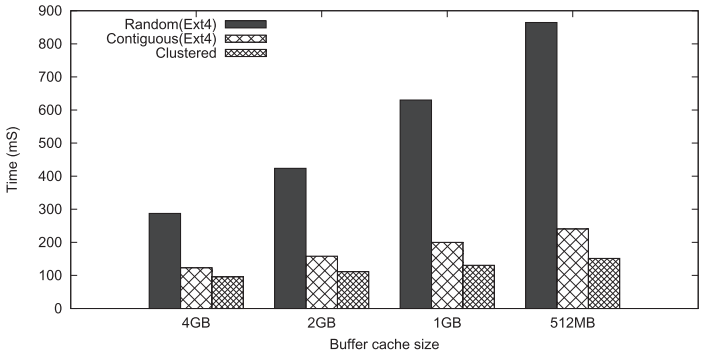
We modified the DiskSim 4.0 simulator [Bucy et al. 2008] to simulate the buffer cache layer with an LRU policy. The DiskSim simulator has been widely used in storage system studies and its accuracy has been extensively validated. The default disk model we used was a Seagate Cheetah 15k.5, which has a capacity of 146GB, a speed of 15000 RPM, and a cache memory of 16MB. For buffer cache simulation, we tested a variety of buffer cache sizes from 512MB to 4GB. Three disk block layouts for the blog contents, each of which reflects a different storage system status, were used for comparison, as follows.

— *Random.* The *Random* layout assumes that the system has run for a while and that files for an article being created are difficult to place on physically contiguous disk space. To reflect this scheme, files for an article are spread over the disk somewhat randomly, with the restriction that blocks for an individual file are located contiguously. The Ext4 filesystem is used for this layout.
— *Contiguous.* The *Contiguous* layout reflects the best case for the Ext4 filesystem in that file blocks for an article are as contiguous as possible under the filesystem (see Wang and Li [2003] for details).
— *Clustered.* In the *Clustered* scheme, file blocks are located according to how cluster-based filesystems operate. File blocks for an article are thus mostly contiguous on the disk.

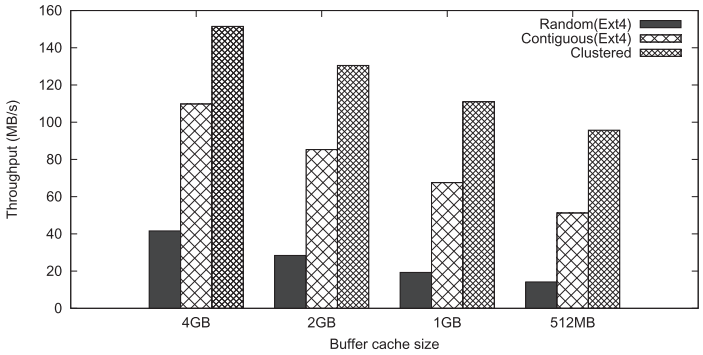
We first examined hit rates for the buffer cache for the Clustered scheme. Over-prefetching due to clustered read operations may result in more or less inefficient

Table IX. Hit Rate for the Buffer Cache

	4 GB	2 GB	1 GB	512 MB
Random	0.55	0.47	0.39	0.33
Clustered	0.53	0.46	0.40	0.34



(a) average processing time for file accesses



(b) throughput

Fig. 19. Performance comparison for different data placement schemes.

utilization of the buffer cache by evicting other file data to be accessed soon; this may harm the overall system performance because buffer cache misses are several orders of magnitude slower than hits [Wachs et al. 2007]. However, we believe that clustering has an insignificant effect on performance in the buffer cache. This is an intuitive conclusion considering our previous observation that only 3.2% of article accesses involve partial transfer, where overprefetching may occur.

Table IX shows that for each cache size, the buffer cache hit rate remains almost the same, regardless of whether the underlying filesystem is Random or Clustered. The results show that even though some file blocks are sometimes evicted too early by overprefetching, the impact of overprefetching on the buffer cache hit rates is negligible.

Another disadvantage of the Clustered scheme is the prolonged transfer time due to the increased unit I/O size. However, we believe that the significant performance improvement from clustering will offset this penalty.

Figure 19 shows the average request processing time and overall throughput for the schemes. Under all three schemes, the larger the buffer cache size, the better are the

throughput and processing time because a larger cache naturally absorbs a greater fraction of the file requests.

As expected, Random shows the worst performance because it does not benefit the locality of file access in the workload and, consequently, induces considerable seek operations.

Even though Contiguous is the ideal filesystem, with file blocks placed as contiguously as possible, it shows inferior performance to the Clustered scheme for two reasons. First, processing fewer and larger article read requests is more efficient than processing a series of small sequential read requests because large-sized file operations amortize positioning delays. Second, when multiple read streams are interleaved, the larger the request size, the fewer seek operations occur. The interleaved series of sequential reads in Contiguous translate to a series of random reads at block device level, and in this case the number of seek operations is determined by the average size of each request.

The Clustered scheme improves processing time for file access by as much as 75.5% and 30.8% on average in comparison to Random and Contiguous, respectively. In terms of throughput, Clustered shows overall improvements of 418% and 60.6% compared to Random and Contiguous, respectively.

7. RELATED WORK

Several studies have focused on the patterns or evolution of user-generated content in Online Social Networks (OSNs) such as blogs. In a study on knowledge-sharing OSNs, Guo et al. [2009] found that posting behavior exhibits strong daily and weekly patterns and that posted content is not generally contributed by a small number of users. Cha et al. [2009] studied information propagation in Flickr and showed that popular photos do not spread widely and quickly throughout the network. Burke et al. [2009] analyzed server log data from Facebook and found that newcomers who see their friends contributing go on to share more content themselves. Moreover, for newcomers who are initially inclined to contribute, receipt of feedback and a wide audience are predictors of increased sharing. Leskovec et al. [2007] studied temporal aspects and topological patterns for information propagation in blogs and found that the popularity of blog articles drops with a power law. In addition, they showed that the size distribution for cascades (number of articles involved) follows a perfect Zipf distribution.

Recently, Duarte et al. [2007] performed comprehensive characterization of blog workloads using HTTP requests collected from a major Brazilian blog service. They presented a few server-side blog workload characteristics including transfer sizes that would be approximated into Pareto distribution with tail index a = 1. The two main differences compared to our result are: (1) Pareto-distributed transfer sizes without filtering out multimedia files and (2) a larger tail index. We conjecture that the difference is largely due to the fact that our blog service tends to accommodate larger size of files such as multimedia files and photographic images.

Though a few findings (e.g., diurnal access patterns) were discussed as blog server characteristics, the primary focus of their study was on understanding a variety of interactions between bloggers and visitors. Specifically, they classified activities in blogs based on blog article postings, comments, and article reads, and used these activities to understand different types of interactions. The study found that more than half of blogs have at least 5% of comment postings for the total visits to the blog, and the aggregate traffic for these blogs does not occupy a large portion of the total blog server traffic. Our work on comprehensive server-side workload characterizations and performance implications in the file system is generally complementary to this study.

At a much lower semantic level, studies have examined the (negative) impact of OSN workload on server performance. Ohara et al. [2009] showed that requests arising from the user contributory nature of OSNs often retrieve and update persistent data. This leads to reduced performance and is a deterrent to the use of multicore architectures. Nagpurkar et al. [2008] found that stalls due to data cache misses are the predominant component of stall cycles. In this regard, we believe that our analysis provides a foundation to guide the design of better server architectures for OSN hosts.

Lastly, popularity evolution models can be developed by examining the long-term popularity evolution of newly generated contents. Borghol et al. [2011] studied the long-term popularity dynamics of user-generated videos by keeping tracking the views to recently uploaded Youtube videos over a eight-month period. They developed a three-phase characterization of popularity evolution for a group of recent videos, and constructed a model for it to be able to generate synthetic workloads for hot set churn statistics, and the evolution of viewing rates and distributions.

8. CONCLUSION

We conducted an empirical analysis of access logs for one of the largest Korean blog hosting sites collected over a period of 12 consecutive days. We investigated user activities on the blog servers and studied behaviors for user-created contents and their distribution patterns. Specifically, we analyzed user activities such as posting and reading of articles, and identified storage usage and network bandwidth consumption for files with different characteristics. We observed access count changes according to lifecycle changes in a blog article and incoming paths to the article.

Our analysis revealed that blog workloads share many characteristics with conventional Web workloads. However, some characteristics, such as the low proportion of image data in the network usage profile, are different from those of conventional Web workloads. In addition, characteristics such as the posting-to-access ratio do not exist for conventional Web workloads. These differences mostly stem from the fundamental nature of blog services, whereby users create most of the content shared with other users.

As an example of applying the findings from our analysis, we proposed a method for improving the performance of a blog server filesystem. Because all article files that belong to the same article are usually read together, our approach uses an article as the unit for read and write operations. Simulation revealed that the suggested approach would lead to a throughput improvement of over 400% in comparison to conventional filesystems.

Besides the suggested approach for blog filesystems, we believe that our results provide valuable information for designers and developers of UCC services that will help in achieving significant performance improvements.

APPENDIX

The Pareto distribution. A distribution is considered heavy-tailed if

$$P[X > x] \backsim x^{-\alpha}, \text{ as } x \to \infty, \ 0 < \alpha < 2.$$

This means that the asymptotic shape of the distribution is hyperbolic, regardless of the behavior of the distribution for small values [Crovella and Taqqu 1999].

The simplest heavy-tailed distribution is the Pareto distribution [Paxson and Floyd 1994], which is hyperbolic over its entire range. The Pareto distribution has the probability mass function

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1},\ \alpha, k > 0,\ x \geq k$$

and the Cumulative Distribution Function (CDF)

$$F(x) = P[X \leq x] = 1 - \left(\frac{k}{x}\right)^{\alpha},$$

where α is the tail index and k is the smallest possible value for the random variable.

To determine whether the CDF fits the Pareto distribution well, we use Log-Log Complementary Distribution (LLCD) plots [Crovella and Bestavros 1997]. The LLCD is transformed from the CDF by plotting $log(1 - F(x))$ on the vertical axis and $log(x)$ on the horizontal axis.

Plotted in this way, heavy-tailed distributions have the property

$$\frac{d\log(1 - F(x))}{d\log(x)} = -\alpha,\ x > \theta$$

for some θ. In practice, we select a θ value from the preceding LLCD which we inspect using an approximately linear slope and then estimate the slope $-\alpha$ using least-squares regression.

The tail index α and the R^2 goodness-of-fit value explain a number of properties of the Pareto distribution. If $0 < \alpha < 2$, then the distribution has infinite variance; if $0 < \alpha \leq 1$, then the distribution has an infinite mean. Thus, as α decreases, an arbitrarily large portion of the probability mass may be present in the tail of the distribution. The approximation process becomes more precise and reliable as R^2 approaches 1.

REFERENCES

ABAN, I. B., MEERSCHAERT, M. M., AND PANORSKA, A. K. 2006. Parameter estimation for the truncated pareto distribution. *J. Amer. Statist. Assoc. 101*, 473, 270–277.

ARLITT, M. F. AND JIN, T. 2000. A workload characterization study of the 1998 world cup web site. *IEEE Netw. 14*, 3, 33–37.

ARLITT, M. F. AND WILLIAMSON, C. L. 1997. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Trans. Netw. 5*, 5, 631–645.

BARFORD, P. AND CROVELLA, M. 1999. A performance evaluation of hyper text transfer protocols. *SIGMETRICS Perform. Eval. Rev. 27*, 1, 188–197.

BENT, L., RABINOVICH, M., VOELKER, G. M., AND XIAO, Z. 2004. Characterization of a large web site population with implications for content delivery. In *Proceedings of the 13th International World Wide Web Conference*.

BORGHOL, Y., MITRA, S., ARDON, S., CARLSSON, N., EAGER, D., AND MAHANTI, A. 2011. Characterizing and modeling popularity of user-generated videos. *Perform. Eval. 68*, 11, 1037–1055.

BUCY, J. S., SCHINDLER, J., SCHLOSSER, S. W., AND GANGER, G. R. 2008. The disksim simulation environment version 4.0 reference manual. Tech. rep. CMU-PDL-08-101, Carnegie Mellon University.

BURKE, M., MARLOW, C., AND LENTO, T. 2009. Feed me: Motivating newcomer contribution in social network sites. In *Proceedings of the 27th ACM CHI Conference on Human Factors in Computing Systems*.

CHA, M., MISLOVE, A., AND GUMMADI, K. P. 2009. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th International Conference on World Wide Web*.

CHALLENGER, J. 1996. A distributed web server and its performance analysis on multiple platforms. In *Proceedings of the 16th International Conference on Distributed Computing Systems*.

CROVELLA, M. E. AND BESTAVROS, A. 1997. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Trans. Netw. 5*, 6, 835–846.

CROVELLA, M. E. AND TAQQU, M. S. 1999. Estimating the heavy tail index from scaling properties. *Meth. Comput. Appl. Probab. 1*, 1, 55–79.

DINGLE, A., MACNAIR, E., AND NGUYEN, T. 1999. An analysis of web server performance. In *Proceedings of the Global Telecommunication Conference*.

DUARTE, F., MATTOS, B., BESTAVROS, A., ALMEIDA, V., AND ALMEIDA, J. 2007. Traffic characteristics and communication patterns in blogosphere. In *Proceedings of the International Conference on Weblogs and Social Media*.

FABER, A. M., GUPTA, M., AND VIECCO, C. H. 2006. Revisiting web server workload invariants in the context of scientific web sites. In *Proceedings of the ACM/IEEE Conference on Supercomputing*.

GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. 2007. Youtube traffic characterization: A view from the edge. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*.

GILL, P., ARLITT, M., CARLSSON, N., MAHANTI, A., AND WILLIAMSON, C. 2011. Characterizing organizational use of web-based services: Methodology, challenges, observations, and insights. *ACM Trans. Web 5*, 4, 19:1–19:23.

GUO, L., TAN, E., CHEN, S., ZHANG, X., AND ZHAO, Y. E. 2009. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

HOLMEDAHL, V., SMITH, B., AND YANG, T. 1998. Cooperative caching of dynamic content on a distributed web server. In *Proceedings of the 7th International Symposium on High Performance Distributed Computing*.

IYENGAR, A. AND CHALLENGER, J. 1997. Improving web server performance by caching dynamic data. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*.

KANT, K. AND WON, Y. 1999. Performance impact of uncached file accesses in specweb99. In *Proceedings of the 2nd IEEE Workshop on Workload Characterization*.

KRISHNAMURTHY, B. 2009. A measure of online social networks. In *Proceedings of the 1st International Conference on Communication Systems and Networks*.

LESKOVEC, J., MCGLOHON, M., FALOUTSOS, C., GLANCE, N., AND HURST, M. 2007. Cascading behavior in large blog graphs. In *Proceedings of the 7th SIAM International Conference on Data Mining*.

LI, Z., CHEN, Z., SRINIVASAN, S. M., AND ZHOU, Y. 2004. C-miner: Mining block correlations in storage systems. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*.

LIMPERT, E., STAHEL, W. A., AND ABBT, M. 2001. *Log-Normal Distributions across the Sciences: Keys and Clues*. BioScience.

NAGPURKAR, P., HORN, W., GOPALAKRISHNAN, U., DUBEY, N., JANN, J., AND PATTNAIK, P. 2008. Workload characterization of selected JEE-based web 2.0 applications. In *Proceedings of the IEEE International Symposium on Workload Characterization*.

OHARA, M., NAGPURKAR, P., UEDA, Y., AND ISHIZAKI, K. 2009. The data-centricity of web 2.0 workloads and its impact on server performance. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*.

OKE, A. AND BUNT, R. B. 2002. Hierarchical workload characterization for a busy web server. In *Proceedings of the 12th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*.

PATTERSON, R. H., GIBSON, G. A., GINTING, E., STODOLSKY, D., AND ZELENKA, J. 1995. Informed prefetching and caching. In *Proceedings of the 15th ACM Symposium on Operating System Principles*.

PAXSON, V. AND FLOYD, S. 1994. Wide-area traffic: the failure of poisson modeling. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*.

RODRIGUEZ, P. 2009. Web infrastructure for the 21st century. In *Proceedings of the 18th International World Wide Web Conference*.

SHRIVER, E., GABBER, E., HUANG, L., AND STEIN, C. A. 2001. *Proceedings of the USENIX Annual Technical Conference*.

STEWART, C., LEVENTI, M., AND SHEN, K. 2008. Empirical examination of a collaborative web application. In *Proceedings of the IEEE International Symposium on Workload Characterization*.

TOMKINS, A., PATTERSON, R. H., AND GIBSON, G. 1997. Informed multi-process prefetching and caching. In *Proceedings of the ACM SIGMETRICS Conference*.

VERES, S. AND IONESCU, D. 2009. Measurement-Based traffic characterization for web 2.0 applications. In *Proceedings of the International Instrumentation and Measurement Technology Conference*.

WACHS, M., ABD-EL-MALEK, M., THERESKA, E., AND GANGER, G. R. 2007. Argon: Performance insulation for shared storage servers. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*.

WANG, J. AND LI, D. 2003.A light-weight, temporary file system for large-scale web servers. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*.

WILLIAMS, A., ARLITT, M., WILLIAMSON, C., AND BARKER, K. 2005. *Web Workload Characterization: Ten Years Later*. Springer.

ZIPF, G. K. 1949. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley.

Received September 2011; revised May 2012; accepted August 2012