Characterizing Caching Workload of a Large Commercial Content Delivery Network

M. Zubair Shafiq
The University of Iowa
Iowa City, IA
Email: zubair-shafiq@uiowa.edu

Amir R. Khakpour
Verizon Digital Media Services
Los Angeles, CA
Email: amir.khakpour@verizon.com

Alex X. Liu
Michigan State University
East Lansing, MI
Email: alexliu@cse.msu.edu

***Abstract*—Content Delivery Networks (CDNs) have emerged as a dominant mechanism to deliver content over the Internet. Despite their importance, to our best knowledge, large-scale performance analysis of CDN cache performance is lacking in prior literature. A CDN serves many content publishers simultaneously and thus has unique workload characteristics; it typically deals with extremely large content volume and high content diversity from multiple content publishers. CDNs also have unique performance metrics; other than hit ratio, CDNs also need to minimize network and disk load on cache servers.**

In this paper, we present measurement and analysis of caching workload at a large commercial CDN serving thousands of popular content publishers. Using detailed logs from four geographically distributed CDN cache servers, we analyze over 600 million content requests accounting for more than 1.3 petabytes worth of traffic. We analyze CDN workload from a wide range of perspectives, including request composition, size, popularity, and temporal dynamics. Using real-world logs, we also evaluate cache replacement algorithms, including two enhancements designed based on our CDN workload analysis: N-hit and content-aware caching. The results show that these enhancements achieve substantial performance gains in terms of cache hit ratio, disk load, and origin traffic volume.

I. Introduction

Motivation. The Internet has experienced a massive traffic growth for many years and this growth is projected to continue in future. Internet traffic volume per capita is expected to triple between 2013-2018, reaching 15 gigabytes by year 2018 [1]. It is noteworthy that a few top content publishers account for a majority of the Internet traffic. For instance, Netflix and YouTube account for more than half of the peak downstream traffic in North America [2]. In order to improve QoS by bringing content closer to their end users, most major content publishers such as Facebook and Netflix employ third-party Content Delivery Networks (CDNs) [3]. According to Cisco, 55% of the Internet traffic and 67% of all video traffic will cross CDNs by year 2018 [1].

Background. A CDN operator typically owns or rents spaces from multiple data centers, called Points of Presence (PoPs), which are geographically distributed around the world. Figure 1 illustrates CDN architecture and demonstrates how CDNs deliver content to end users. When a content publisher contracts a CDN to host their content, a user request to their content (such as a web page, an image, or a video file) is redirected to the closest cache server inside a PoP via DNS redirection, HTTP redirection, URL rewriting, or anycast [3]. If that caching server does not have the content, then the server fetches it from the origin server, sends it to the user, and also caches the content in storage. Thus, for later requests to the same content, the server directly sends its cached copy to the users if the Time-To-Live (TTL) value has not expired. CDNs provide many benefits such as faster content delivery to end users, higher service availability, better user experience, less traffic to content publisher servers, and better protection of content publishers from Denial of Service (DoS) due to their larger capacity of handling legitimate flash crowds and malicious attack traffic.

Why Study CDN Caching? Content caching is one of the most important functions of a CDN. Researchers have extensively studied web proxy caching [4], [5]; however, *there is lack of prior work on characterization and performance evaluation of real-world CDN workloads and content caching strategies used by CDNs.*

• **Unique Workload.** CDNs serve many content publishers simultaneously, their workload has unique characteristics, which have implications for content caching strategies. First, the volume of content for caching is extremely high, especially with the increasing amount of user-generated content in online social networks. Second, the diversity of content for caching is extremely high, and possibly includes all types of digital content from many different content publishers. Third, the temporalness of content accessing behavior is extremely strong and flash crowds are a commonplace.

• **Unique Performance Metrics.** CDNs also have unique performance metrics as compared to other forms of caching. Hit ratio is commonly used by researchers evaluating caching strategies. In addition to hit ratio, CDN caching systems have

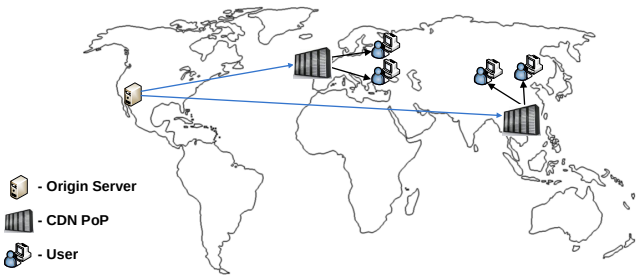


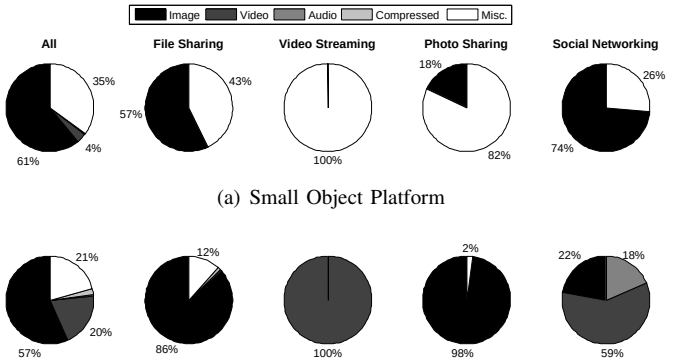
Fig. 1. Architecture of a Content Delivery Network (CDN)

two specialized performance metrics, namely disk load and traffic from origin servers. Disk load is a key performance metric in CDN caching systems because it is often the performance bottleneck (comparable to network latencies) during peak usage time. Reducing disk load directly reduces I/O delay and further reduces end-user latency. Note that each millisecond counts for CDNs – a few milliseconds disadvantage may put one CDN operator behind others and can result in loss of traffic/contracts with content publishers. Origin traffic volume is another key factor for CDN caching systems because fewer bytes from the origin implies less infrastructure and transit traffic costs [6].

Proposed Research. In this paper, we conduct a measurement study of a large commercial CDN which serves thousands of popular content publishers. We have collected detailed traces from four different geographically distributed CDN cache servers. Our data set contains over 600 million content requests accounting for more than 1.3 petabytes worth of traffic. We analyze CDN workload from a wide range of perspectives, including request composition, size, popularity, and temporal dynamics. Our analysis reveals several characteristics of CDN workload and cache performance. For instance, object popularity is highly skewed and exhibits significant variations over time and across different content types. Using real-world CDN workload, we evaluate the well-known least-recently-used (LRU) and least-frequently-used (LFU) cache replacement algorithms. The results show that LRU achieves mean hit ratios of 84% and 92% for the small and large object caches, respectively. Comparing LRU and LFU, we observe that LRU consistently outperforms LFU by 5-10% in terms of hit ratio. Since CDNs have unique workload characteristics and performance metrics [7], [8], we explore the following CDN-specific enhancements to LRU/LFU.

• First, motivated by the skewed content popularity observed in CDN workloads, we implement and evaluate **N-hit caching**. Skewed content popularity distribution is well-known for web server workloads [9]. Several solutions have been proposed to exclude tail content from web caches for improving hit ratio [5]. CDN operators are concerned about hit ratio as well as excessive disk I/O load caused by tail content. The basic idea of N-hit caching is to use a counting Bloom filter for efficiently keeping track of the request frequency of objects. Without N-Hit caching, an object will be cached after the first request; whereas with N-Hit caching, it will be cached upon the N+1 th request. Our results show that N-hit caching with LRU/LFU *reduces disk load by more than 90%* for both small and large object caches without affecting hit ratio.

• Second, motivated by the diverse workload characteristics of different content types, we implement and evaluate **content-aware caching**. Cache partitioning is widely used in shared memory caches to account for diverse concurrently executing applications [10]–[12]. Following the same concept, the basic idea of content-aware caching is to dynamically partition the total cache space into sub-caches of different sizes where different sub-caches are used to cache objects of different types and this partition dynamically evolves over time. Our results show that content-aware caching with LRU/LFU *reduces cache misses by 40% and 73%* for the small and large object caches, respectively.



(a) Small Object Platform

(b) Large Object Platform

Fig. 2. Request (count) composition for content publisher categories

II. Workload Characteristics

For this study, we collected real-world traffic traces from a large-scale commercial CDN service. Next, we first present our data collection methodology. Then we analyze several CDN workload characteristics in our data set.

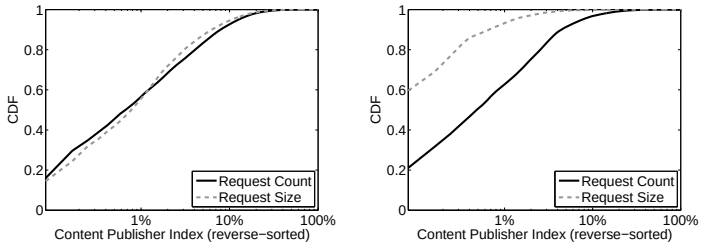
A. Background & Data Collection

The CDN studied in this paper uses two platforms (where a platform refers to the infrastructure of a suite of hardware servers and software services), called small and large object platforms, which are optimized to handle small objects (of size typically less than 300 KB) and large objects (of size typically larger than 300 KB), respectively. Content publishers decide which objects should be cached on which platforms based on their performance metrics and cost models, although the CDN operator offers guidelines on assigning objects to the two platforms. The small object platform is optimized to store small objects (which are mostly small web objects such as HTML, CSS, and image thumbnails) and provides comparably higher performance (with higher pricing). The large object platform is optimized to store large objects (which are mostly videos, high resolution images, software) and provides comparably lower performance (with lower pricing).

We collected anonymized logs of content requests from four major cache servers of the CDN service over the duration of one week. Table I provides summary statistics of our data set. In total, the data set consists of more than 600 million

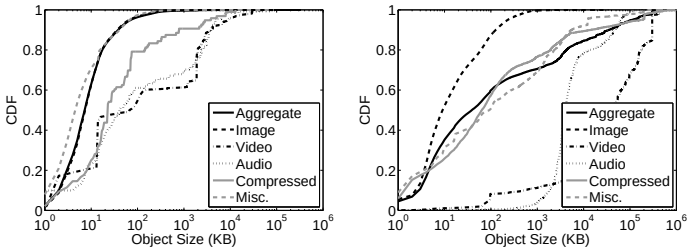
TABLE I
Data set summary statistics

Server Geographical Location	**# Requests** (millions)	**Request Size** (terabytes)
Australia	110.77	420.86
Europe	113.01	149.86
Eastern United States	125.39	141.91
Western United States	262.65	606.95
Total	611.82	1,319.58



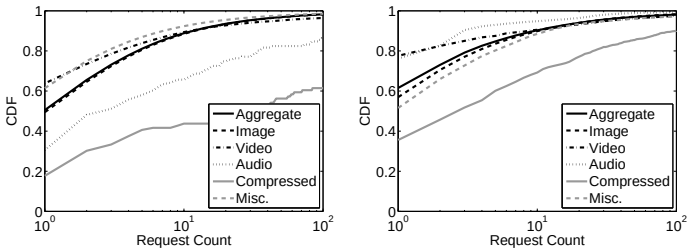
(a) Small Object Platform (b) Large Object Platform

Fig. 3. Distributions of request count and request size



(a) Small Object Platform (b) Large Object Platform

Fig. 4. Object size distributions



(a) Small Object Platform (b) Large Object Platform

Fig. 5. Request count distributions

content requests accounting for more than 1.3 petabytes worth of traffic. The four PoPs are located in Australia, Europe, eastern United States and western United States. We obtained separate logs for both small and large object platforms. All personally identifiable information in our data set (*e.g.*, IP addresses) has been anonymized to protect the privacy of end users without affecting the usefulness of our results. Each record in our logs contains the content publisher's identifier, hashed URL, query string, object file type, object size in bytes, and the time stamp when the request was received. The objects are categorized into `image`, `audio`, `video`, `compressed`, and `misc` based on their file extensions. For example, `image` category includes .jpeg, .png, .gif, .tiff, .bmp, etc. extensions and `video` category includes .flv, .mp4, .mpg, .avi, .wmw, etc. extensions. A substantial fraction of streaming video content in our data set uses chunk-based video streaming. The chunks of a video file (e.g., byte ranges) are treated as individual objects by the CDN operator.

B. Request Composition

We begin by analyzing content requests with respect to content publishers. Our data set includes content requests from a diverse mix of websites, e.g., social networking, video streaming, photo sharing, file sharing, etc. We do not reveal the names of content publishers or their websites due to proprietary restrictions; however, several of these content publishers are in the Alexa top 100 list. Figure 2 compares the request composition for different categories of content publishers. Overall, request composition seems to match the type of content publisher. For instance, the large object platform is used by video streaming websites solely to cache video files and photo sharing websites for image files. Social networking websites use the large object platform to cache a mix of image, video, and audio files. In contrast, the small object platform is primarily used to cache thumbnail images and HTML files by all content publisher categories. More specifically, `image-misc-video` breakdown for the small object platform is 61-35-4. For the large object platform, this breakdown changes to 57-21-20.

Figure 3 plots the cumulative distribution functions (CDFs) of request count (*i.e.*, the total number of content requests) and request size (*i.e.*, the total size of requested objects) across content publishers. We observe that content requests exhibit strong skewness across content publishers, i.e., a small fraction of content publishers are responsible for a majority of content requests. More specifically, top 1% content publishers account for up to 60% of the total request count for both small and large object platforms. Moreover, top 1% content publishers account for more than 90% of the total request size for the large object platform.

C. Object Size

We next investigate object sizes across different content types. Figure 4 plots the CDFs of object sizes for small and large object platforms. Overall, we observe that object sizes in the small object platform are much smaller than those in the large object platform. This is expected because both content publishers and the CDN operator prefer to use the large object platform to store large objects and the small object platform to store small objects due to both performance and cost reasons.

Across content types in Figure 4, we note that size distributions of `audio` and `video` content exhibit modality, unlike other content types. In the small object platform, objects in the `image` and `misc` categories, of sizes mostly less than 100 KB, have the smallest sizes compared to objects in other categories. Overall, since the `image` category dominates others in terms of the number of objects, the aggregate distribution closely follows that of images. For the large object platform, the majority of the objects in the `image` and `misc` categories have sizes less than 100 KB, and the objects in the `audio` category are mostly a few megabytes in size. The sizes of `video` objects vary over a wide range, from a few megabytes to hundreds of megabytes, where smaller video sizes mostly represent byte-range requests for video file chunks.

D. Object Popularity

We now investigate individual object popularity, which is quantified in terms of request count. Figure 5 plots the CDFs of request count for different content types. We observe the expected long-tail distributions for different content types and

both platforms. The distributions highlight that a small fraction of objects appear in most requests and most objects are requested very infrequently. Object popularity distributions for the large object platform have longer tails than those for the small object platform. For example, more than 60% and 50% objects in the large and small object platforms, respectively, are requested only once. This observation indicates that the objects in the small object platform are generally "more cacheable" than those in the large object platform. We also observe significant variations across different content types for both platforms. The ordering of lines for different content types is not consistent across both platforms. For example, `audio` content has the longest tail for the large object platform, whereas it has the second-shortest tail for the small object platform.

E. Temporal Dynamics

To understand how content requests change over time, we first analyze the timeseries of request counts. Figure 6 shows the timeseries of request counts for different content types in both small and large object platforms. We observe that all timeseries exhibit diurnal variations, with minor variations across different content types. Overall, the request volume peaks from morning till late evening and bottoms out after mid-night. The timeseries of `audio` and `compressed` content are spiky because of their lower volumes. The request volume of `video` and `compressed` content substantially increases on Friday for the large object platform, while other content types exhibit regular diurnal patterns.

To further analyze the temporal dynamics of object popularity, we analyze the timeseries of object uniqueness for different content types. Object uniqueness reveals temporal locality of objects by counting how many keys do not repeat [13], [14]. We define *object uniqueness ratio* as the ratio of the total number of unique object requested to the total number of requests. Its value is in the range $(0, 1]$, where the values closer to 0 indicate that some popular objects are repeatedly accessed and the values closer to 1 indicate that most objects are accessed infrequently. Figure 7 shows the timeseries of object uniqueness ratio for all content types in both platforms. The aggregated timeseries of object uniqueness ratio exhibits diurnal variations between 0.1 and 0.3, which indicates that object popularity changes over time. Moreover, diurnal dips represent relatively more temporal locality. Comparing Figures 7 and 6, we observe that high request volume generally corresponds to low object uniqueness ratio. This observation implies that popular objects are accessed more frequently during peak times.

It is interesting to note the differences in object uniqueness ratios across content types in Figure 7. For example, `misc` content, containing essential web content such as HTML and JavaScript, consistently has the lowest object uniqueness ratio as compared to other content types for both platforms. `Video` content also closely follows the `misc` category because the time-driven dissemination of video content (such as TV episodes and popular videos) leads to frequent requests for popular videos in a short time span. The remaining content types generally have higher object uniqueness ratios. In particular, the `image` content, which is also the largest contributor in terms of request volume, has the average object uniqueness ratios of approximately 0.3 and 0.2 for small and large object platforms, respectively.

III. Cache Performance Analysis

The performance of caching schemes is typically evaluated in terms of hit (or miss) ratio. However, CDNs have distinct performance metrics as compared to other forms of caching. In addition to hit ratio, CDN caching systems have two specialized performance metrics, namely disk load and origin traffic volume. These metrics are are not necessarily correlated with each other, e.g., improving hit ratio does not always mean reduction in disk load or traffic from origin servers. Using trace-driven simulation based on CDN workloads, we evaluate various cache replacement algorithms in terms of the following three metrics.

1. *Hit Ratio*: It is defined as the number of requests that are served CDN cache (without fetching from origin servers) divided by the total number of requests. This metric is in the range of $[0, 1]$, where values closer to 1 indicate better caching performance.

2. *Disk Load*: It is defined as the number of disk operations that the CDN performs in processing and serving requests from the storage media.

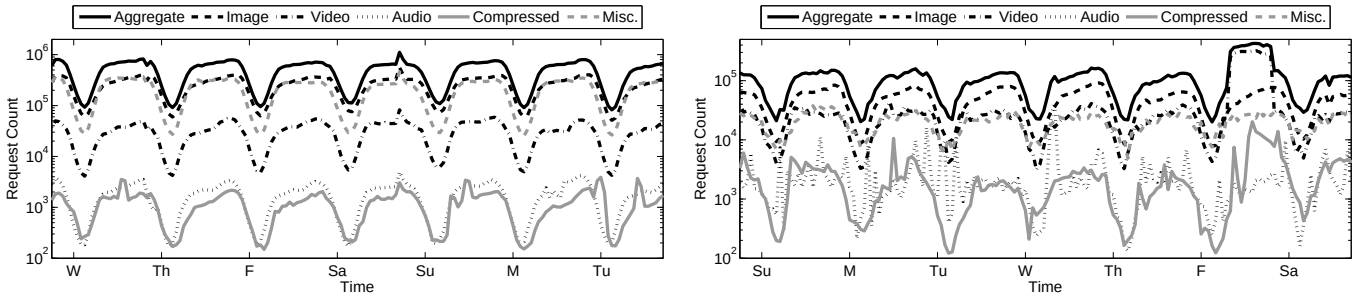
3. *Origin Traffic Volume*: It is defined as the total number of bytes fetched from the origin servers of content publishers.

This section presents a thorough evaluation of various cache replacement algorithms; in particular, how cache performance relates to time and different content types for small and large object platforms.

A. LRU/LFU

We start our analysis of CDN cache behavior by evaluating the performance of the well-known least-recently-used (LRU) and least-frequently-used (LFU) cache replacement algorithms for comparison. We conduct trace-driven simulations for both LRU and LFU running as the caching replacement algorithm in a CDN PoP. Our evaluations are based on a finite cache size of 100 gigabytes for the small object platform and 1 terabytes for the large object platform. These cache sizes are representative of typical configurations used by the CDN operator in production systems.

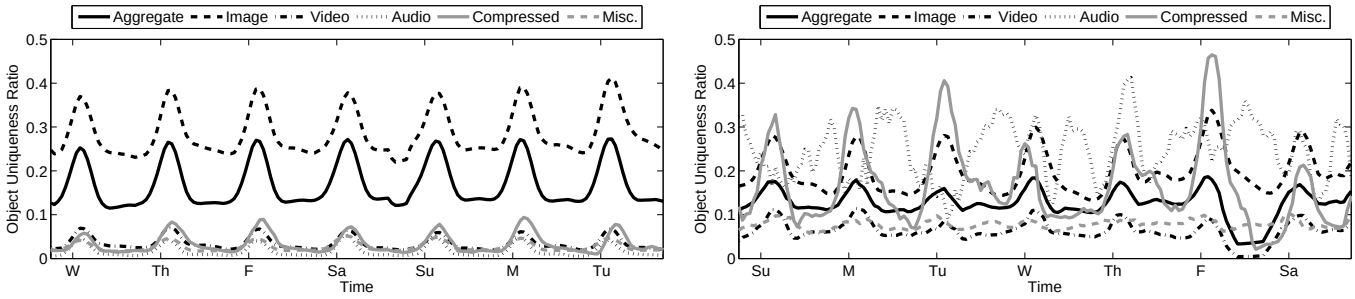
Overall, our results demonstrate that LRU consistently outperforms LFU for all performance metrics and for both platforms. For instance, we observe that LRU outperforms LFU by 5-10% in terms of hit ratio. Due to space constraints, we focus on results for LRU based caching in rest of the paper. Figure 8(a) shows the performance of LRU for the small object platform. For each metric, we show the performance of LRU for aggregate (*i.e.*, all) requests as well as for requests of each content type (*i.e.*, `image`, `audio`, `video`, `compressed`, and `misc`). We have several main observations. First, for aggregate requests, all performance



(a) Small Object Platform

(b) Large Object Platform

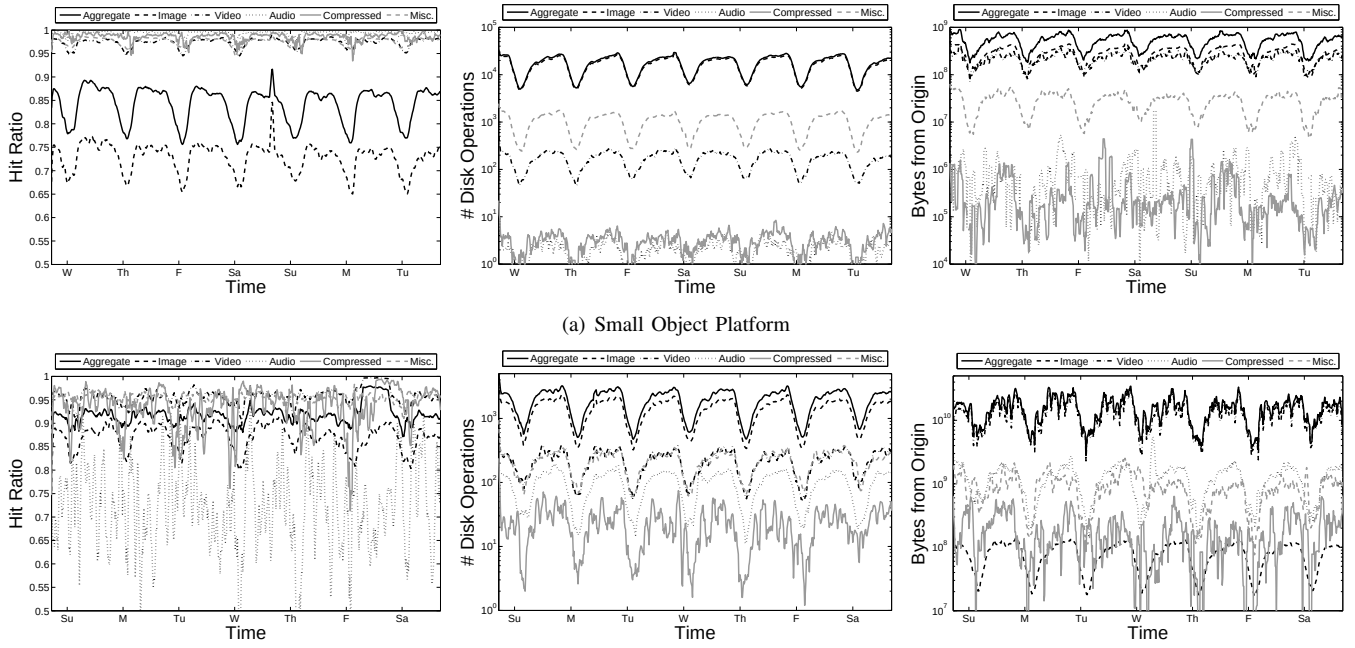
Fig. 6. Timeseries of object request volume



(a) Small Object Platform

(b) Large Object Platform

Fig. 7. Timeseries of object uniqueness



(a) Small Object Platform

(b) Large Object Platform

Fig. 8. LRU caching performance for small and large object platforms

metrics show strong diurnal variations. Specifically, hit ratio bottoms out after midnight and peaks in morning and stays flat in the afternoon. Disk load and origin traffic volume exhibit similar trends. Second, the caching performance of different content types exhibits disparity. For example, the hit ratios of `misc` and `video` are up to 95% whereas the hit ratio of `image` content varies between 70%-90%. Furthermore, only `image` content accounts for more than 90% of all requests for disk load and origin traffic volume.

Figure 8(b) shows the performance of LRU for the large object platform. Similarly, for each metric, we show the performance of LRU for aggregate requests as well as for requests of each content type. We have several key observations. First, for aggregate requests, the diurnal variations are not as smooth as we observed for the small object platform. We observe diverse diurnal variations across content types. Second, similar to our observation from the small object platform, `video` and `misc` get better hit ratios as compared to the other types.

For example, hit ratios of misc and video generally remain above 90% whereas the hit ratio of image again hovers between 70%-90%. For the large object platform, image accounts for more than 90% of disk load and video accounts for 90% of origin traffic volume.

Comparing hit ratio with object uniqueness ratio in Figure 7, we note that the hit ratio performance is strongly correlated with object uniqueness measurement; specifically, smaller object uniqueness ratios lead to higher hit ratios. This makes intuitive sense as only repeating requests result in cache hits. Moreover, the number of bytes from origin in the large object platform is orders of magnitude larger as compared to the small object platform. This makes intuitive sense as the sizes of the objects in the large object platform are significantly larger than the sizes of the objects in the small object platform.

Several extensions of LRU/LFU have been proposed and evaluated in prior web caching literature (see [5] and references therein). These extensions specialize the caching strategies to target specific application scenarios and traffic behavior. Since CDNs have unique workload characteristics and performance metrics [7], [8], we next discuss CDN-specific enhancements to LRU/LFU.

B. N-Hit Caching

Recall from Figures 3 and 5 that the object popularity distribution exhibits strong skewness for both platforms. The tail of the skewed popularity distribution contains the objects that may occupy cache but would not contribute much towards cache hits. CDN operators are concerned about hit ratio as well as excessive disk I/O operations (which often result in hundreds of milliseconds processing delay in the worst case) caused by the tail content. For instance, Figure 9 plots the correlation between cache response delay (excluding network delays) and 95th-percentile traffic load. We note that cache response delay increases significantly during peak traffic load, indicating potential disk I/O related delays during peak load.

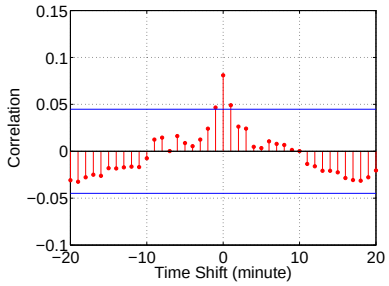


Fig. 9. Correlation between cache response delay and 95th-percentile traffic load

Skewed content popularity distribution is well-known for web server workloads and several solutions have been proposed in prior art [5], [9]. Motivated by the skewed content popularity observed in CDN workloads, we implement and evaluate N-hit caching. In typical caching schemes, an object is cached upon the first request; however, based on this observation, our idea is to filter out unpopular objects and do not cache them to better utilize finite CDN storage space for popular objects. Specifically, in N-Hit caching, within a certain time interval, the first N requests to an object do not cause the object to be cached (*i.e.*, they are ignored from the caching perspective), and only the $N+1$ th request causes the object to be cached. Although each object contributes up to $N+1$ cache misses, no matter they are popular or not, overall N-Hit caching improves hit ratio because the chance of more popular objects getting swapped out by less popular objects is reduced.

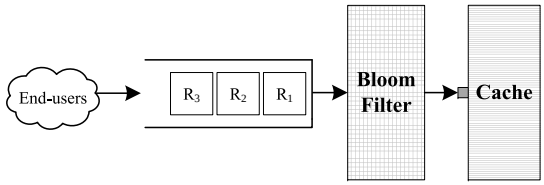
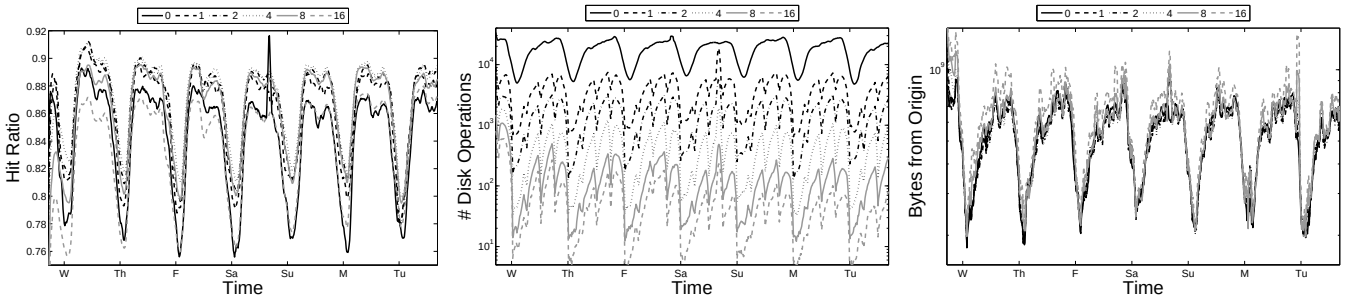


Fig. 10. N-hit caching

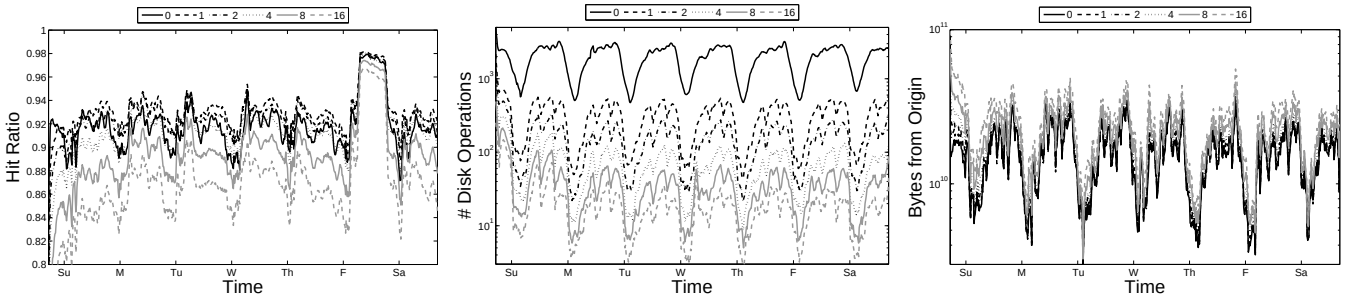
To efficiently track the request counts for a large number of objects, we use a counting Bloom filter [15], [16]. Figure 10, where R_1, R_2, and R_3 represent object requests by end users, illustrates the approach of counting Bloom filter based N-hit caching. Bloom filters offer significant advantage over other data structures in terms of space and time complexity, while answering membership queries with a tunable low probability for false positives [17]. A counting Bloom filter consists of an array A of m counters where the number of bits in each counter is b and the initial value of each counter is 0, and k independent hash functions $h_1, h_2, ..., h_k$ whose outputs are uniformly distributed in the range $[1, m]$. To test whether an element e is in a counting Bloom filter A, we check the value of $A[h_i(e)]$ for every $1 \leq i \leq k$; if there exists at least one $1 \leq i \leq k$ such that $A[h_i(e)] = 0$, then e is not in A; otherwise, e is in A with a high probability. To insert an element e into a counting Bloom filter, we increment $A[h_i(e)]$ by one for every $1 \leq i \leq k$. To delete an element e from a counting Bloom filter, we decrement $A[h_i(e)]$ by one for every $1 \leq i \leq k$. These counts are incremented when a new element is inserted, and decremented when an element is deleted.

The false negative rate of a counting Bloom filter is always 0. The false positive rate of a counting Bloom filter is determined by the number of counters m, the number of hash functions k, and the number of inserted elements n [15], [18]. Both larger m and k lead to smaller false positive rates. On the other hand, larger n leads to larger false positive rates. In our experiments, we conservatively select values of k and m to keep the false positive rate negligible. In particular, we select $k = 10$ and $m = 100,000,000$, which is at least two orders of magnitude more than peak request volume. Furthermore, to account for monotonically increasing n, we periodically reset the counting Bloom filter. The lower bound on Bloom filter reset period can be obtained from the distribution of object inter-arrival times. Given that the average inter-arrival times of more than 90% objects are less than 6 hours, we set the Bloom filter reset period to 6 hours in our evaluation.

We evaluate the performance of LRU with N-hit caching for both small and large object platforms. In our experiments, we vary the value of N to be 0, 1, 2, 4, 8, and 16. For the counting Bloom filter, we choose $m = 100$ million, $k = 10$, and $b = 4$. Figure 11 shows the performance of LRU with N-hit caching for the small and large object platforms. We note that the highest hit ratio and the smallest origin traffic volume



(a) Small Object Platform



(b) Large Object Platform

Fig. 11. N-hit LRU caching performance for small and large object platforms. Note that N-hit caching provides orders of magnitude reduction in disk operations while providing similar or better hit ratio and origin traffic volume.

are observed when $N = 4$, whereas disk load decreases monotonically as N increases. More specifically, 4-hit caching improves the caching performance of LRU by increasing its hit ratio from 84.0% to 86.1% (equivalent to 13% reduction in cache misses), *decreasing disk load by* 99%, and decreasing origin traffic volume by 5%. For the large object platform, the highest hit ratio and the smallest number of bytes from origin are observed when $N = 2$, whereas disk load again decreases monotonically as N increases. More specifically, 2-hit caching improves the caching performance of LRU by increasing its hit ratio from 91.8% to 92.5% (equivalent to 8% reduction in cache misses), decreasing disk load by 92%, and decreasing origin traffic volume by 8%.

C. Content-aware Caching

Recall that content request patterns exhibit strong temporal dynamics for different content types. Thus, we implement a content-aware caching approach to dynamically cache different types of objects accordingly. Cache partitioning is used in shared memory caches to account for diverse concurrently executing applications [10]–[12]. Following the same concept, the idea of content-aware caching is to dynamically partition the total cache space into sub-caches of different sizes where different sub-caches are used to cache objects of different types and this partition dynamically evolves over time.

Figure 12 illustrates the approach of content-aware caching. Let $\tau_i(t) \in [0, 1]$ denote the fractional size of sub-cache i at time t, where $i \in \{1, 2, 3, 4, 5\}$ is an indicator variable that represents the five content types: `image`, `video`, `audio`, `compressed`, and `misc`. Thus, we have $\sum_{i=1}^{5} \tau_i(t) = 1$. For better caching performance, we would like to allocate more space to the sub-caches of the content types that will lead

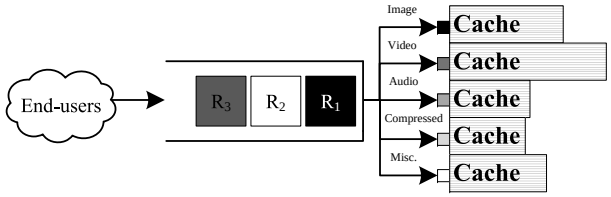
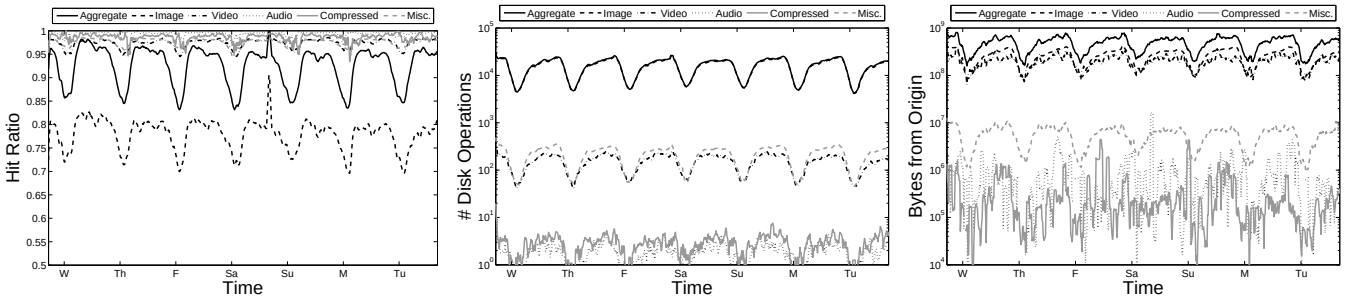


Fig. 12. Content-aware caching

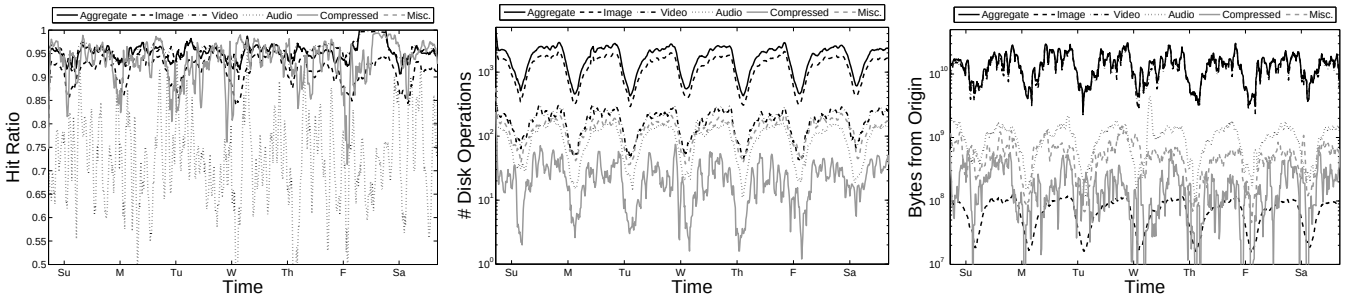
to larger request counts and better caching performance in near future. Towards this end, let $\alpha_i(t) \in [0, +\infty)$ denote the metric that jointly quantifies the request counts and caching performance for content type i at time t. We experimented with different candidate metrics for α and observed that hit count (*i.e.*, the number of objects served from cache) and hit size (*i.e.*, the size of objects served from cache), are most suited for small and large object platforms, respectively. Hit count turns out to be the best candidate for the small object platform because this platform deals with an order of magnitude more request count than the large object platform. Likewise, hit size is the best candidate for the large object platform because this platform deals with objects that are orders of magnitude larger in size compared to those of the small object platform. Given $\alpha_i(t)$, we can compute $\tau_i(t)$ as:

$$\tau_i(t) = \frac{\alpha_i(t)}{\sum_{\forall i} \alpha_i(t)}.$$

However, the value of $\alpha_i(t)$ is not known in advance. Therefore, we have to predict its value at time t using the available history at time $t - 1$. Autoregressive Integrated Moving Average (ARIMA) [19] is suitable to handle non-stationary data. To predict $\alpha_i(t)$ from its available history, we model its timeseries using ARIMA. An ARIMA(p,d,q) process modeling α_i is defined as follows:



(a) Small Object Platform



(b) Large Object Platform

Fig. 13. Content-aware LRU caching performance for small and large object platforms

$$(1-\sum_{j=1}^{p}\phi_j B^j)(1-B)^d\alpha_i = z_t(1+\sum_{k=1}^{q}\theta_k B^k),$$

where z_t denotes the white noise with variance σ^2 and B is the backward shift operator. The first component on the left side (ϕ_j) models $\alpha_i(t)$ as a linear combination of its p historical observations. The second component on the left side takes d differences to convert a non-stationary time series to a stationary one. The component on the right side is a linear combination of current noise and q historical noises. To fit ARIMA model to our data, we conduct an exhaustive search over different values of p, d, and q parameters. We vary p and q in the range $[0, 100]$ and d in the range $[0, 10]$. For each (p,d,q) tuple, we estimate the model coefficients $\phi_1, \phi_2, ..., \phi_j, \theta_1, \theta_2, ..., \theta_k$, and σ using the maximum likelihood method. We compute the Akaike Information Criterion (AIC) statistic and test the residuals of each model for i.i.d. using sample autocorrelation function (ACF), partial autocorrelation function (PACF), Ljung-Box statistic, McLeod-Li statistic, Jarque-Bera statistic, turning points test, and rank test [19]. Finally, we select the model that has the minimum AIC value and passes most residual tests.

We estimate separate models of α_i for all content types and re-tune these models after regular intervals. More frequent updates after small intervals (*e.g.*, order of seconds) make the cache partitions unstable. Less frequent updates after large intervals decrease the adaptive benefit provided by cache partitioning. To ensure that partitions stabilize, we selected the value of this interval to be 15 minutes for both small and large object platforms after pilot experimentation.

Figure 13 shows the performance of LRU with content-aware caching for both platforms. We show the performance of LRU with content-aware caching for aggregate requests as well as for requests of each content type. In comparison with the performance of LRU, we have the following key observations. First, for the small object platform, content-aware caching improves the caching performance of LRU by increasing its hit ratio from 84.0% to 90.3% (equivalent to 40% reduction in cache misses), decreasing disk load by 17%, and decreasing origin traffic volume by 14%. Second, for the large object platform, content-aware caching improves the caching performance of LRU by increasing its hit ratio from 91.8% to 97.8% (equivalent to 73% reduction in cache misses), decreasing disk load by 16%, and decreasing origin traffic volume by 5%. Across different content categories, content-aware caching improves LRU performance for `image`, `audio`, and `compressed` content for the small object platform. Recall from Figure 6 that `audio` and `compressed` content types having much smaller request volume than others. Since content-aware caching provides separate sub-caches for different content types, objects belonging to low request volume content types do not get pushed out of the cache by the objects of high request volume content types.

IV. Related Work

Prior work on CDN performance measurement and workload characterization fall into two categories: active measurement and passive measurement. Some active measurement studies on CDNs have been done [20]–[25]; however, they are small-scale studies due to the active nature and they do not provide a characterization of content requests other than those they injected. Some passive measurement studies on CDNs have been done [26], [27]; however, they are limited to analyzing traffic from CoralCDN, a free and open CDN based on the PlanetLab testbed, whose workload is orders of magnitude smaller than that of large commercial CDNs.

Active Measurements. These studies actively probe CDNs to obtain measurements, such as using ping packets to measure latency and file downloading to measure throughput. These studies are limited to active latency and throughput measurements, and they do not focus on CDN caching performance. In [20], Krishnamurthy *et al.* conducted active client-end measurements to study CDN performance. In [21], Huang *et al.* developed a distributed platform based on PlanetLab to measure the delay performance of CDNs. They used this platform to measure the performance of Akamai and Limelight CDNs. In [22], Su *et al.* studied the performance of audio and video streaming services provided by Akamai using active measurements. In [25], Adhikari *et al.* conducted active bandwidth measurement of three CDNs used by Netflix. They examined instantaneous bandwidth variations of these CDNs from different vantage points. In a similar study [24], Adhikari *et al.* conducted active measurement study of three CDNs used by Hulu. In [23], Triukose *et al.* used DipZoom measurement platform to find and probe Akamai edge servers from hundreds of vantage points.

Passive Measurements. In [26], Freedman analyzed the operational performance of CoralCDN. In [27], Wendell and Freedman analyzed flash crowds in CoralCDN. The main limitation of these studies is that they do not closely reflect the characteristics of large commercial CDNs because CoralCDN is a small CDN based on the PlanetLab testbed with orders of magnitude smaller traffic load than popular commercial CDNs. Saroiu *et al.* used passively collected traffic traces from the border router of a university campus [28]. They compared Akamai and general HTTP traffic with P2P traffic belonging to Gnutella and Kazaa. They studied the utility of local proxy cache against the caching services provided by Akamai. Our work differs from this study in that we conducted our measurement on the CDN side whereas they conducted their measurement on the end-user side. We see all requests handled by CDN cache server whereas they only see the requests from a university. Thus, our study is at a much larger scale in terms of the number of requests, unique objects, and content publishers.

V. Conclusion

This paper presents a measurement study of a large commercial CDN serving thousands of content publishers. We analyzed CDN workload from a wide range of perspectives. Our empirical analysis revealed many characteristics of CDN workload and shed light on various ways to improve CDN cache performance. We found that N-hit caching is best at reducing disk load (up to 99% reduction) whereas content-aware caching is best at improving hit ratio (up to 73% reduction in cache misses) and origin traffic volume (up to 14% reduction). We also evaluated content-aware N-hit caching: we filter out unpopular objects using a counting Bloom filter and partition the total cache into sub-caches for different content types. The results demonstrate that content-aware N-hit caching achieves simultaneous gains for all performance metrics.

References

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2013-2018," June 2014.

[2] "SandVine: Global Internet Phenomenon Report 1H-2014."

[3] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. Maggs, S. Uhlig, V. Aggarwal, and F. Schneider, *Collaboration Opportunities for Content Delivery and Network Infrastructures*, H. Haddadi and O. Bonaventure, Eds. Recent Advances in Networking, ACM SIGCOMM, 2013.

[4] J. Wang, "A Survey of Web Caching Schemes for the Internet," *ACM SIGCOMM CCR*, vol. 29, no. 5, pp. 36–46, 1999.

[5] S. Podlipnig and L. Bszrmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.

[6] S. Hasan, S. Gorinsky, C. Dovrolis, and R. K. Sitaraman, "Trade-offs in Optimizing the Cache Deployments of CDNs," in *IEEE Infocom*, 2014.

[7] G. Hasslinger and O. Hohlfeld, "Efficiency of Caches for Content Distribution on the Internet," in *International Teletraffic Congress*, 2010.

[8] G. Hasslinger and K. Ntougias, "Evaluation of Caching Strategies Based on Access Statistics of Past Requests," in *International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems*, 2014.

[9] M. F. Arlitt and C. L. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631–645, 1997.

[10] H. S. Stone, J. Turek, and J. L. Wolf, "Optimal Partitioning of Cache Memory," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1054–1068, 1992.

[11] G. E. Suh, L. Rudolph, and S. Devadas, "Dynamic Partitioning of Shared Cache Memory," *Journal of Supercomputing Architecture*, vol. 28, no. 1, pp. 7–26, 2004.

[12] M. K. Qureshi and Y. N. Patt, "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches," in *IEEE/ACM International Symposium on Microarchitecture*, 2006.

[13] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal Locality in Today's Content Caching: Why it Matters and How to Model it," *ACM SIGCOMM CCR*, 2013.

[14] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," in *ACM SIGMETRICS/Performance*, 2012.

[15] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[16] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An Improved Construction for Counting Bloom Filters," in *14th Annual European Symposium on Algorithms*, 2006.

[17] B. H. Bloom, "Space/time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[18] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, "On the False-positive Rate of Bloom Filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.

[19] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, G. Casella, S. Fienberg, and I. Olkin, Eds. Springer, 2001.

[20] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the Use and Performance of Content Distribution Networks," in *ACM SIGCOMM Workshop on Internet Measurement*, 2001.

[21] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and Evaluating Large-Scale CDNs," in *ACM IMC*, 2008.

[22] A.-J. Su and A. Kuzmanovic, "Thinning Akamai," in *ACM IMC*, 2008.

[23] S. Triukose, Z. Wen, and M. Rabinovich, "Measuring a Commercial Content Delivery Network," in *ACM WWW*, 2011.

[24] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z.-L. Zhang, "A Tale of 3 CDNs: An Active Measurement Study of Hulu and Its CDNs," in *IEEE Global Internet Symposium*, 2012.

[25] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, M. Steiner, V. Hilt, and Z.-L. Zhang, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery," in *IEEE Infocom*, 2012.

[26] M. J. Freedman, "Experiences with CoralCDN: A Five-Year Operational View," in *USENIX/ACM NSDI*, 2010.

[27] P. Wendell and M. J. Freedman, "Going Viral: Flash Crowds in an Open CDN," in *ACM IMC*, 2011.

[28] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," in *ACM Symposium on Operating Systems Design and Implementation*, 2002.