Computer Networks 107 (2016) 94–103

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

DBit: Assessing statistically significant differences in CDN performance

Zahaib Akhtar[a,*], Alefiya Hussain[b], Ethan Katz-Bassett[a], Ramesh Govindan[a]

[a] Computer Science Department, University of Southern California, United States
[b] University of Southern California Information Sciences Institute (ISI), United States

ARTICLE INFO

Article history:
Received 17 April 2016
Revised 18 May 2016
Accepted 29 May 2016
Available online 2 June 2016

Keywords:
Content distribution networks
Performance analysis methodologies

ABSTRACT

As the volume of content served by content distribution networks (CDNs) grows, these networks evolve to improve performance. Their performance is difficult to characterize because it depends on a number of factors. In this paper, we develop a methodology called DBit that can determine whether one CDN's user-perceived performance is statistically different from another. We validate DBit and demonstrate its usefulness on CDNs used for photo delivery. We use PlanetLab to collect HTTP download data for 14.5 million photo fetches and 5 million video fetches and RIPE Atlas nodes hosted in end-user homes in 1470 ASes worldwide to obtain 470,400 photo fetches respectively, from three Photo CDNs and two Video CDNs. We find that DBit can identify significant performance differences not just between CDNs, but also across time and location.

1. Introduction

Beyond Akamai's early deployment of a large content distribution network (CDN), other content providers have recently built out their own CDNs [1–5]. Today, a very large proportion of Internet traffic is served by CDNs, so it is important to develop robust methodologies to understand their performance. The performance of CDNs depend on many factors, including placement of the front-end servers in the topology, the quality of their connectivity to clients, the cache sizes and the cache efficacy, the compute capacity of the front-ends, the network connectivity between the front-ends and the back-end, the efficiency of the storage system (e.g., for photos and videos) or the compute system (e.g., for search) at the back-end, and so forth.

There is, however, a lack of a widely accepted systematic methodology for their performance analysis. Existing techniques for CDN performance comparison fall into two classes. Some studies use first order statistics (mean, median, percentiles) of performance measures and compare CDNs based on the differences in the magnitude of these first-order statistics [6–8]. Others [9] compare CDNs based on the distribution of the difference in performance metrics across all clients. In either case, whether the performance difference is significant is often a matter of judgement.

Contributions. In this paper, we make the following contributions: (1) we show that comparisons based on first or second order statistics can lead to faulty conclusions, (2) we consider a complementary methodology for comparing CDNs based on *statistically significant* differences in the *distributions* of performance metrics (like latency) as seen by multiple clients. Specifically, given two CDNs **A** and **B**, we ask: *how can we systematically determine if **A**'s user-perceived performance is statistically better than **B**'s, or vice versa?* Our approach, called DBit, can answer this question for a continuous-valued performance metric such as latency or throughput. In this paper, we focus on latency comparisons.

The key idea behind DBit is to use standard hypothesis testing to establish statistical significance, but adapted for CDN architectures. Suppose we measure multiple samples of CDN download latency, for two CDNs **A** and **B**, from each client in a large set of CDN clients. At each client, its samples approximate the distribution of latency seen for each CDN. The key idea behind DBit is to use a test of differences in the distribution at each client: this determines whether, at that client, **A** is statistically better than **B**. Then, we use another statistical test to determine if there is a statistically significant number of clients at which **A** is significantly better than **B**.

Using active measurements of 14.5 million photo fetches and 5 million video fetches spanning over a period of eight months from PlanetLab and RIPE Atlas, we show: (a) that DBit signals statistical differences where these might be expected to exist (such as diurnal differences, or differences between cache fetch and CDN backend fetch performance); (b) that when DBit detects a statistical difference, the magnitude of the differences are significant (several hundred milliseconds) and systematic (visible across multiple, geographically dispersed) vantage points; and (c) that it is flexible enough to be used to perform a variety of other comparisons, such as tail latency performance and outlier detection.

* Corresponding author.
E-mail addresses: zakhtar@usc.edu (Z. Akhtar), hussain@isi.edu (A. Hussain), ethan.kb@usc.edu (E. Katz-Bassett), ramesh@usc.edu (R. Govindan).

http://dx.doi.org/10.1016/j.comnet.2016.05.020
1389-1286/

Use cases. DBit is a first step towards a systematic methodology for assessing statistical differences between CDN performance. By itself, it can either be used by third-party companies (like Conviva or Keynote [10,11]) to provide a comparative performance of major CDNs. Customers can use such an assessment as input in deciding to use a CDN service. Moreover, a CDN can use this methodology to compare itself against its competitors (for whom it will not have direct access to performance metrics), to help focus its engineering efforts.

2. Motivation and design rationale

In this section we discuss the point in the design space which DBit aims to occupy and our rationale for choosing it. CDNs in general are dynamic entities with a great degree of diversity in the way they are engineered and the characteristics of their workload. For instance, a CDN with a highly uniform workload like Facebook can be engineered through very specific optimizations driven by the characteristics of its workload as compared to a third party CDN like Akamai [5,12]. Similarly, factors such as scale, cache size, cache efficacy, network connectivity to the backend and clients etc. also affect CDN performance. As a result, CDN performance is known to vary greatly across both temporal and spatial dimensions [10]. Therefore, any straightforward attempt to do CDN comparison is likely to fall short, since care is required in selection of the correct metrics and the methodology [6].

Unfortunately, there is a lack of a widely accepted methodology to perform CDN comparison. Current techniques range from back of the envelope calculations or anecdotal evidence to schemes which are either application specific, or use first and second order statistics to compare proxy metrics such as DNS resolution time [6,13]. It is, therefore, not surprising that deciding whether the performance difference between CDNs is significant is often left to judgment.

Our design of DBit is motivated by the need for a systematic methodology for CDN comparison. We place two requirements on the design of DBit: that it declares one CDN to be better than another only when there is a statistically significant performance difference; and that it be flexible and can accommodate a variety of performance comparisons.

In the following section we discuss the design of DBit and also explain our design choices in light of the design principles.

3. How DBIT works

In this section we describe the internals of DBit. Suppose we want to determine whether the latency difference between CDN **A** and CDN **B** is statistically significant. Conceptually, DBit has three distinct stages:

Stage 1: DBit obtains active measurements to CDNs from a set of vantage points $\mathcal{V}$. Each active measurement from a node v in $\mathcal{V}$ to **A** or **B** produces one sample of the performance metric of interest (e.g., latency or throughput). Suppose we model this performance metric as a random variable $X_{A,v}$ and $X_{B,v}$ for the two CDNs at each node v. In general, the distributions of X may not be known a priori and can be different for different CDNs, because the corresponding performance metric may depend upon many factors, such as, the location of the client and Photo CDN load, competing network traffic, etc.

Stage 2: DBit looks for statistically significant *distributional differences* in the empirical distributions of the random variables X. We ask: is the *distribution* of $X_{A,v}$ statistically better or worse than $X_{B,v}$? If we use latency as the performance metric then "better" implies "faster" and the above questions translates to: is CDN **A** faster than CDN **B**?

We frame the above question as a hypothesis, namely that **A** is faster than **B**, which can then be answered using hypothesis testing. For this purpose, we use the **two-sample one-sided** Kolmogorov–Smirnov test (or K–S test) [14]. The KS-Test is a well-known non-parametric test and makes no assumptions about the underlying distribution, hence making it a good fit for our purpose. Its decision is based on the KS-statistic which is the maximum difference between the two cumulative distributions. Fig. 1(a) shows two sample distributions for **A** and **B**, the arrow shows the KS-statistic (maximum distance between the CDFs).

Given the distributions of $X_{A,v}$ and $X_{B,v}$ we are now in a position to test our hypothesis that **A** is *faster* than **B** against the null hypothesis that **A** is *similar or slower* than **B** for a given *significance level*. The significance level indicates a degree of confidence in the verdict. Smaller significance levels are better. For example, a significance level of 0.05 (used in this paper) means that the null hypothesis can be rejected with 95% confidence.

The output of the KS-test is a single bit $b_{A,B,v}$ for each vantage point v which is 1 (respectively 0) if the null hypothesis could (respectively could not) be rejected in the favor of our alternate hypothesis. Thus, if $b_{A,B,v}$ is 1, it implies that at some part of the distribution **A** is faster than **B** for vantage point v at the specified significance level. However, the KS-test is one-sided, which leaves the possibility that **B** is faster than **A** in some other part of the distribution. Fig. 1(b) shows an example of this ambiguity.

To deal with this ambiguity, DBit also tests the converse hypothesis, namely that **B** is faster than **A**, before making a final decision. It generates a final decision bit $c_{A,B,v}$ which is 1 if and only if the hypothesis that **A** is faster than **B** is true and the hypothesis **B** is faster than **A** is false or vice versa. At the end of the second stage, each vantage point generates a bit $c_{A,B,v}$: these bits are used as input to the third state.

Stage 3: Finally, DBit uses the Binomial test [15] to determine if the fraction of nodes with $c_{A,B,v}$ bits being 1 is statistically significant. If, for a significant number (as determined by the Binomial test) of nodes, $c_{A,B,v}$ is 1, then we conclude that **A** is indeed faster than **B**. The fraction of $c_{A,B,v}$ bits being 1 required to determine statistical significance depends on the number of vantage points used (Fig. 1(c)): for example, if the total number of vantage points is 250 then *at least* 150[1] vantage points with $c_{A,B,v}$ equal to 1 are required to establish statistical significance.

Choice of statistical test. While we choose the KS-test, it is not the only test that can be used to detect statistical differences between distributions: other tests such as the Anderson–Darling test or Chi-Squared test can also be used. However, these tests either require a priori assumptions about the underlying distribution or are only designed to test whether a given sample comes from a specific probability distribution. These constraints are contrary to the DBit's generality principle which requires that minimum assumptions be made about the data. Hence, the non-parametric K–S test is more suited for comparing empirical distributions the way DBit does. As an aside, the K–S test is known to have flaws, for example when testing whether a given distribution matches the normal distribution [16], but this does not apply in our setting since we use the two-sample one-sided version of the test to compare two samples rather than testing whether a sample matches the normal distribution.

Why simpler approaches are insufficient. Comparing CDNs using means or percentiles can be misleading [17,18] in at least two cases: when CDN performance measures like latency have skewed distributions (as is often the case with CDNs serving clients with significant geographic diversity); or when the distributions are multi-modal (e.g., when vantage points are directed to

[1] The value 150 is calculated as 60% of 250.

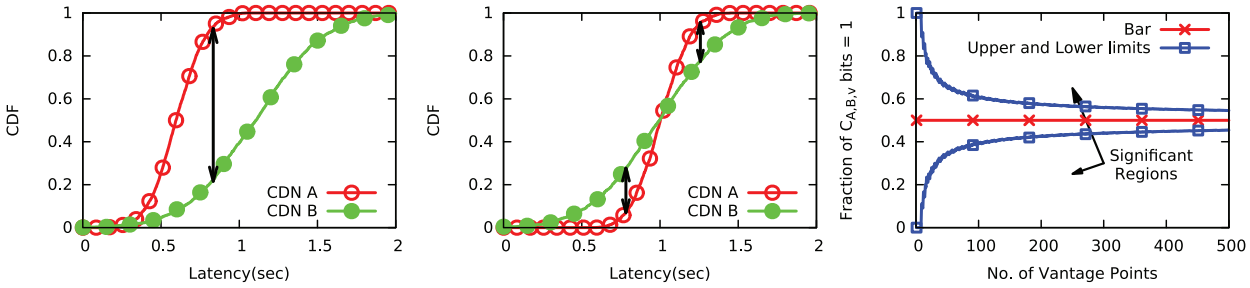

Fig. 1. (a) A sample plot showing that CDN A is faster than CDN B. KS-statistic is shown by arrow. (b) A sample plot showing the ambiguous case. (c) The significant regions show the fraction of 1's required for a 95% confidence in the Binomial test verdict.

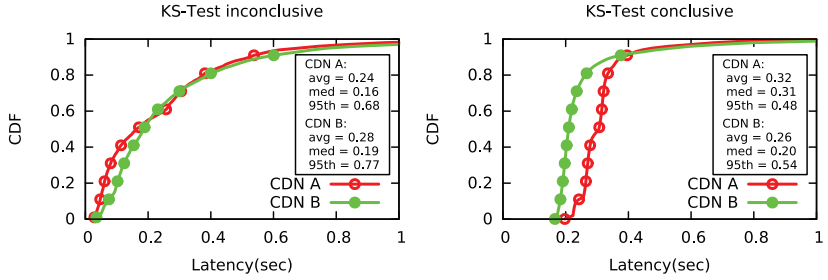

Fig. 2. (a) The latency distributions of two CDNs for which the KS-Test is inconclusive, but the first order statistics show CDN A is better than CDN B. The reason KS-Test is inconclusive because the distribution for CDN A is bimodal, notice that between 60th and 70th percentile CDN A performs worse than CDN B which is enough evidence for KS-Test to be inconclusive. (b) The latency distributions of two CDNs for which the KS-Test is conclusive, but the first order statistics are ambiguous as CDN A is worse than CDN B with respect to average and median but better at the 95th percentile. The reason KS-Test is not inconclusive because the difference in performance at the tail is insignificant evidence for KS-Test, whereas the performance difference between 0 and 90th percentile is significant.

geographically different front end servers due to load balancing). For example, Fig. 2(a) shows the CDF of HTTP request completion latencies from two different CDNs, where the KS-Test is inconclusive but the first order statistics show that CDN A is consistently better than CDN B. Similarly, Fig. 2(b) shows a different example where KS-Test is conclusive but first-order statistics are inconclusive. Notice that CDN A is worse than CDN B with respect to average and median but better at the 95th percentile.[2] In scenarios such as these, using first order statistics can lead to erroneous conclusions, and comparing distributions is often the most robust way to judge statistical differences.

4. Data collection methodology

In this section we describe our methodology to collect data which we later use to demonstrate how DBit's approach can be applied to detect statistically significant performance differences. The data is collected through active measurements spanning a period of six months. We use two different testbeds to fetch content from popular Photo CDNs and record the download latency.

Testbed selection. In order to collect data we use two well known testbeds, namely PlanetLab and RIPE Atlas. We fully acknowledge the shortcomings of both PlanetLab and RIPE Atlas testbeds in terms of location of their vantage points and their suitability for measuring the performance of CDNs. For instance, it is well known that PlanetLab nodes on average tend to have better connectivity than real users in the wild. This is largely due to the fact that 85% of PlanetLab nodes are located in research organizations or university campuses which typically have better provisioned networks with high speed upstream connections [19]. Moreover, user accounts on a PlanetLab node run inside a virtualized environment, hence any latency measurements that involves the NIC (Network Interface Card), such as HTTP request to fetch an object from an external source can simply get inflated due to the virtualization overhead and the load imposed by concurrently active user accounts on a particular node. Similarly, the RIPE Atlas nodes are also known to be biased towards the research community, although a number of nodes are hosted in end user homes but these nodes are mostly setup by researchers using the testbed.

Addressing bias in vantage point selection is a non-goal of this paper. Our goal is to demonstrate how DBit can be used to compare two CDNs, given measurements from several vantage points. In our case, we have used two open measurement infrastructures for this demonstration. While our actual results might possibly be skewed by vantage point selection bias, the value of our paper is in showing how to use DBit to study various aspects of CDN performance. A study or service that claims to compare CDNs would have to address this bias, but once they do, they can leverage the statistical validity of DBit's hypothesis tests. DBit is also not limited by scale, since the data is collected from distributed vantage points, it can easily scale to bigger measurement studies, in fact many real world systems routinely collect such data, as discussed more in Section 6.

The CDNs our primary data set was collected from three Photo CDNs: Google+, Facebook and Flickr. The general architecture of these three Photo CDNs is as follows: Photo CDNs direct client photo fetches to *front-end servers*, and a cache miss results in an access to a *photo back-end*. Each photo is accessed by a URL. We obtained direct CDN URLs to photos by employing user facing APIs for each of the Photo CDN (Facebook's Graph API, Google's Data API, Flickr's Data API.)

Google+: Recent work has shown that Google has expanded its web serving infrastructures and directs search requests to satellite front-ends, which relay the requests to back-end data centers [4]. Through measurements spanning the Google address space we have found that these satellite front-ends also serve requests for Google+ photos. Therefore, Google+ has front-ends at 1400 distinct sites across the world [4].

Facebook: Facebook uses its own set of cache front-end servers [5] and also relies on Akamai [20] for front-end servers to

[2] Both these examples are taken from real-world measurements described later.

serve photos around the globe. In our paper, we treat the two sets of cache servers differently, since they may have different performance properties [21]. Hence, in what follows, every reference to the Akamai CDN refers to *Facebook's use of Akamai*. Beyond nine known Facebook front-end servers (the edge caches in [5]), we have discovered through active probing 14 additional sites belonging to Facebook. Of these additional sites, 1 is in the US and the remaining are in Europe and Asia.[3]

Flickr: An analysis of DNS names from our measurements indicates Flickr directs clients to three photo back-ends. These sites host three of the five known Yahoo! data centers [22]. To study CDN performance using a range of different Internet applications, we also collect data from two popular Video CDNs, namely Facebook and YouTube. Video content delivery differs widely from photo or other file delivery. Most Video CDNs serve video using HTTP adaptive chunking protocols such as HTTP Live Streaming (HLS) or HTTP Dynamic Streaming (HDS) which allow the video players to dynamically switch bitrates depending on the user bandwidth and playback buffer occupancy. Video QoE metrics are therefore, also much more complex than those used for photos or file delivery. Previous work on video QoE modelling has shown that the user perceived quality of a video sessions depends on a number of factors such as (1) *join time*: the time it takes for playback to start, (2) *average bitrate*: the session life time average of the bitrates played, (3) *rebuffering events*: the video player going into a buffering state due to playback buffer drainage and (4) *bitrate change frequency*: the rate of bitrate switches in a session.

Delivering high video QoE not only depends on good CDN performance but also relies on the effectiveness of the adaptive bitrate algorithms (ABR) used by video players and as such is an active area of research. Since our goal in this paper is to show how DBit can be used to measure the performance of CDNs and not to measure the effectiveness of ABR algorithms, we use a slightly different metric for Video CDNs : namely the time to first byte (TTFB) of the first video chunk. The TTFB is a direct indicators of the CDN performance since it shows how quickly a CDN is able to start serving a request. It should be noted that TTFB also impacts the join time, in particular, if TTFB is high then the join time will be inflated as well.

Vantage points. To collect HTTP download data from Photo CDNs, we use 162 PlanetLab vantage points each from a distinct site. PlanetLab provides rich visibility into components of performance and allows us to record the latency of each stage of the entire life-cycle of a HTTP request.

We also use the RIPE Atlas testbed to collect data from a more constrained environment. RIPE does not have the flexibility of PlanetLab since it does not provide direct access to the RIPE probes. Moreover, measurements once configured and launched from the RIPE dashboard can't be changed dynamically. RIPE is also limited in reporting the performance of different stages of a HTTP request, its current version only reports the download completion time but gives no insight into other stages such as the time to perform the TCP handshake or the time to first byte etc. However, it still provides us with a valuable resource in demonstrating DBit's ability to work with a constrained dataset. We select a set of 1470 RIPE vantage points, each in a distinct AS, which are hosted in end user homes. Our video measurements were solely collected from the PlanetLab testbed using the 162 vantage points.

Latency measurements and reference stream. From each Planetlab vantage point, we measure two forms of latency: *cold-fetch latency* which is the time taken to download a non-cached photo and *hot-fetch latency* which is the time taken to download a cached (accessed in recent past) photos. All downloads were performed through cURL using direct CDN URLs to both photos and videos. cURL provides separate latency measures for DNS resolution, TCP connection setup, time-to-first-byte (the time between sending the HTTP request, and receiving the first byte) and time to download the entire photo.

To measure hot-fetch latency each vantage point repeatedly downloads a photo every 30 min, and for cold fetch latency a vantage point downloads a different photo every hour. We assign 150 different photos to each vantage point. This number ensures that a photo will be re-downloaded after six days which is long enough for the photo to be evicted from any cache. We have confirmed this using our measurements: for example, on Facebook we found the eviction times to be between 52 and 60 h.

Our methodology for videos followed a similar approach. In particular a vantage point repeatedly requested the same video every 30 min for hot fetches and requested a different video every one hour for cold fetches.

We followed substantially the same methodology for the RIPE vantage points, but with one important difference: all accesses from RIPE vantage points were to photos that were of resolution 1×1.[4] Since most RIPE nodes are in home networks, we used smaller photos to minimally disturb home users' perceived network performance. Moreover, because of limitations in the way measurement campaigns can be mounted on RIPE, we were only able to measure hot-fetch latencies from the RIPE vantage points. We do not perform video fetches from PlanetLab nodes due to the aforementioned concerns.

Photos reference stream. To measure these forms of latency, we uploaded a total of 600,000 photos, or 200,000 for each of the three providers (Facebook, Google+ and Flickr). We uploaded these to newly created accounts and set privacy settings to ensure that these photos could not be accessed by other users of these Photo CDNs. All photos were JPEG images of the same resolution (960×640) and the same JPEG quality factor (97).

Videos reference stream. To measure the hot and cold fetches for vidoes, we uploaded a total of 75,000 videos to Facebook and YouTube. As with photos, privacy settings were employed to prevent unauthorized access to videos. Our uploaded videos were 2 s long which is sufficient to measure TTFB.

All our Planetlab experiments ran between April and November 2014 during which we retrieved a total of 14.5 million photo fetches and 5 millions video fetches. Our latency measurements for photos from RIPE Atlas ran for 12 days between Dec 1 and 12, 2014.

5. Evaluation

In this section we use our collected data to demonstrate different use cases for DBit. First, we validate the correctness of DBit by checking whether it can detect known performance differences or not. Second, we show how to use DBit to uncover statistically significant difference in performance of a CDN and finally we explore the range of questions which can be answered by DBit. In summary, following are our goals for the evaluation of DBit:

- Does DBit correctly identify differences which are known to be (likely) true and hide differences known not to exist?
- How can DBit be used to detect statistically significant performance differences?
- What is the actual magnitude of differences when DBit identifies statistically significant performance difference?
- Is DBit flexible enough to perform other kinds of comparisons?

[3] Seattle, Amsterdam, Paris, Frankfurt, Hong Kong, Kuala Lumpur, London, Lulea (Sweden), Madrid, Milan, Tokyo, Sao Paulo, Singapore, Vienna

[4] a 1×1 resolution JPG image comprises of a single pixel

Table 1
Hot fetch and cold fetch comparison. DBit correctly identifies that hot fetches have lower latencies than cold fetches.

Hot vs. cold fetches								
	Akamai		Facebook		Flickr		Google+	
	C ∧ H	H ∧ C	C ∧ H	H ∧ C	C ∧ H	H ∧ C	C ∧ H	H ∧ C
$c_{A,B,v}$ **frac.**	0	0.9864	0.034	0.7007	0	0.9116	0	0.9796
P-value	1.1e−44	1.2e−40	3.0e−33	8.1e−08	1.1e−44	1.1e−29	1.1e−44	5.9e−39
Significant	**H** ✓		**H** ✓		**H** ✓		**H** ✓	

Table 2
Time of day comparison for hot fetches, DBit correctly identifies that **P2** has lower latencies than **P5**.

Time of day variation detected by DBit								
	Akamai		Facebook		Flickr		Google+	
	P5 ∧ P2	P2 ∧ P5	P5 ∧ P2	P2 ∧ P5	P5 ∧ P2	P2 ∧ P5	P5 ∧ P2	P2 ∧ P5
$c_{A,B,v}$ **frac.**	0.0370	0.8519	0.1111	0.5741	0.0556	0.7407	0.1111	0.3889
P-val	1.6e−13	1.4e−07	3.3e−09	3.4e−01	2.9e−12	5.4e−04	3.3e−09	1.3e−01
Significant	**P2** ✓		✗		**P2** ✓		✗	

5.1. Is DBit correct?

We first evaluate DBit by testing its ability to detect statistical differences in settings where intuition suggests there should be a significant performance difference.

Hot vs. cold fetches. It is reasonable to expect that hot fetches are faster than cold fetches. To test whether DBit's results are consistent with this expectation, we obtain the distribution of hot and cold fetches $X_{H,v}$ and $X_{C,v}$ for each Photo CDNs at each node v in the set $\mathcal{V}$ and use them as inputs to second stage of DBit. (As an aside, although in Section 3 we described DBit in terms of comparing two CDNs, DBit can be used to compare the performance of a given CDN under different conditions. This is done by conditioning the distributions appropriately, as we have done here by generating separate hot and cold fetch distributions.)

Table 1 shows the results from DBit. For example, the Akamai column shows two hypotheses: C ∧ H and H ∧ C, where ∧ sign denotes a "faster than" relation, hence, H ∧ C should be read as "hot is faster than cold". The $c_{A,B,v}$ row shows the output of the second stage of DBit, that is, total fraction of nodes where the hypothesis is true, so under the hypothesis C ∧ H, at no node is the hypothesis true resulting in a value of 0. Conversely, the high value under the hypothesis H ∧ C signifies that an overwhelming majority of the nodes found that the hot fetches are indeed faster than cold fetches. The P-value row shows the output of DBit's third stage, that is, the probability of getting the fractions in the $c_{A,B,v}$ row due to mere chance. Observe that the values are negligibly small, ruling out the possibility of chance with high probability. By similarly examining other columns, we see that this statistical difference persists across other Photo CDNs. Thus, we can conclude that, *DBit correctly identifies that for all Photo CDNs, hot fetches are distributionally faster than cold fetches*. Finally, the "significant" row shows the final outcome of DBit.

Peak vs. off-peak hours. Internet traffic and server load is known to follow diurnal patterns. Thus, one might expect CDN performance to be better during off-peak hours and worse during peak hours. We use DBit to see if its results are consistent with this expectation. To understand this, we select a subset of 54 Planetlab nodes from our set which lie in the same time zone and divide the 24 h day into 4 h periods (labeled **P1** to **P6**), with **P1** from 2:30 a.m. to 6:29 a.m., **P2** from 6:30 a.m. to 10:29 a.m. and so on (these periods are determined by local time at the corresponding nodes' timezone). For each Photo CDN, we obtain the distribution of each period $X_{P1,v}$, $X_{P2,v}$, $X_{P3,v}$, $X_{P4,v}$, $X_{P5,v}$ and $X_{P6,v}$ for each Photo CDNs at each node v in the set $\mathcal{V}$ and use DBit to compare distributions from one period to the Photo CDN's own distribution in another period for both hot and cold fetches. DBit detects that there is a statistical performance difference between **P2** and **P5**. We show the results in Table 2 for the comparison between **P2** and **P5**. **P2**, the early morning hours between 6:30 a.m. and 10:29 a.m., sees better performance than evening (6:30 –10:29 p.m.) of **P5**. While all CDNs exibit better performance during **P2**, DBit does not determine the differences to be statistically significant in case of Facebook and Google+, since these difference are small.

5.2. Using DBit to compare CDNs

We now use our collected dataset to demonstrate how DBit can be used to compare the performance of CDNs. Towards this end we use data collected from both Photo CDNs and Video CDNs . For photos, we compare the performace of Flickr with Facebook and Google+ and for videos we compare the performance of Facebook with YouTube.

Photo CDN comparison. Flickr uses Yahoo's CDN to serve content around the globe. It is known from previous work that the scale of Yahoo's CDN is much smaller than Akamai and Google [22]. Our measurement of Facebook's content serving infrastructure also confirms that Facebook operates a greater number of Edge Caches than Flickr. Given this background knowledge, it is reasonable to expect that the latency performance of Flickr should be inferior to other Photo CDNs with respect to both hot and cold fetches.

It is known from previous work that CDN performance can vary considerably across both spatial and temporal dimensions [23]. Therefore, in order to apply DBit we divide our PlanetLab vantage points on the basis of continents, which is also the finest granularity permitted by our data. Another study which has greater number of vantage points may choose to group vantage points at much more finer granularities such as cities or metro areas.

We then apply DBit to these two groups as well the global set of vantage points. When comparing Flickr to a given Photo CDN we frame two hypothesis to test both sides of the comparison. For example, when comparing with Facebook, the framed hypothesis are "Facebook is faster than Flickr" and "Flickr is faster than Facebook". Table 3 shows the results of both the second and third stage of DBit for Flickr in comparison to other Photo CDNs for both hot and cold fetches. The $c_{A,B,v}$ row shows the total fraction of nodes where the hypothesis is true, whereas the p-values show if the fraction of

Table 3
Hot and cold fetch results for Flickr against other Photo CDNs. Ticks denote a significant comparison, The coverage includes the total set of nodes, nodes in North America and nodes in Europe.

		Global hot fetch comparison						Global cold fetch comparison					
		Flickr vs. Facebook		Flickr vs. Google+		Flickr vs. Akamai		Flick vs. Facebook		Flickr vs. Google+		Flickr vs. Akamai	
		FB ∧ Fl	Fl ∧ FB	G+ ∧ Fl	Fl ∧ G+	Ak ∧ Fl	Fl ∧ Ak	Fb ∧ Fl	Fl ∧ FB	G+ ∧ Fl	Fl ∧ G+	Ak ∧ Fl	Fl ∧ Ak
Global	$c_{A,B,v}$ **frac.**	0.6481	0.1296	0.9444	0	0.8086	0	0.9074	0.0185	0.9691	0.0062	0.8519	0.0123
	P-value	2.0e−04	5.1e−23	6.1e−35	3.4e−49	7.9e−16	3.4e−49	2.1e−28	2.4e−43	3.1e−40	5.6e−47	1.2e−20	4.5e−45
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓		Facebook ✓		Google+ ✓		Akamai ✓	
		N. America hot fetch comparison						**N. America cold fetch comparison**					
N. America	$c_{A,B,v}$ **frac.**	0.7761	0.0149	0.9701	0	0.9104	0	0.9403	0	0.9403	0	0.806	0.0149
	P-value	6.5e−06	9.2e−19	3.1e−17	1.4e−20	1.5e−12	1.4e−20	1.1e−14	1.4e−20	1.4e−20	1.1e−14	4.5e−07	9.2e−19
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓		Facebook ✓		Google+ ✓		Akamai ✓	
		Europe hot fetch comparison						**Europe cold fetch comparison**					
Europe	$c_{A,B,v}$ **frac.**	0.5949	0.2278	0.962	0	0.7595	0	0.8734	0.0253	0.9873	0.0127	0.8734	0.0127
	P-value	1.2e−01	1.3e−06	2.7e−19	3.3e−24	4.2e−06	3.3e−24	5.5e−12	1.1e−20	3.3e−24	2.7e−22	5.5e−12	2.7e−22
	Significant?	✗		Google+ ✓		Akamai ✓		Facebook ✓		Google+ ✓		Akamai ✓	

Table 4
Photo CDN comparison based on average, median and 95th percentile. All values are in seconds, highest values are shown in bold.

		Global hot fetch first order statistics				Global cold fetch first order statistics			
		Akamai	Facebook	Flickr	Google+	Akamai	Facebook	Flickr	Google+
Global	**Average**	0.36	0.49	**0.97**	0.27	0.84	0.65	**2.39**	0.52
	Median	0.17	0.28	**0.50**	0.12	0.53	0.44	**2.26**	0.36
	95 %ile	4.97	1.4	**5.0**	1.37	4.98	1.77	**5.0**	1.81
		N. America hot fetch first order statistics				**N. America cold fetch first order statistics**			
N. America	**Average**	0.33	0.3	**0.87**	0.29	0.72	0.44	**2.19**	0.48
	Median	0.18	0.18	**0.41**	0.13	0.44	0.26	**2.26**	0.3
	95% ile	1.55	0.97	**4.97**	1.45	2.38	1.45	**5.0**	1.6
		Europe hot fetch first order statistics				**Europe cold fetch first order statistics**			
Europe	**Average**	0.36	0.58	**1.01**	0.23	0.93	0.75	**2.37**	0.53
	Median	0.16	0.42	**0.58**	0.1	0.56	0.61	**2.01**	0.4
	95 %ile	5.0	1.39	**5.0**	0.87	**5.1**	1.67	5.0	1.7

Table 5
Hot and cold fetch results for Facebook, Google+ and Akamai against each other Photo CDNs.

		Global hot fetch comparison						Global cold fetch comparison					
		Facebook vs. Akamai		Facebook vs. Google+		Akamai vs. Google+		Facebook vs. Akamai		Facebook vs. Google+		Akamai vs. Google+	
		FB ∧ Ak	Ak ∧ FB	FB ∧ G+	G+ ∧ FB	Ak ∧ G+	G+ ∧ Ak	FB ∧ Ak	Ak ∧ FB	FB ∧ G+	G+ ∧ FB	Ak ∧ G+	G+ ∧ Ak
Global	$c_{A,B,v}$ **frac.**	0.2407	0.4136	0.1667	0.5370	0.1605	0.5555	0.6605	0.1235	0.3457	0.4877	0.0432	0.8642
	P-value	2.5e−11	3.4e−02	1.8e−18	3.9e−01	3.5e−19	1.8e−01	5.4e−05	7.4e−24	1.1e−04	8.1e−01	1.8e−37	3.4e−22
	Significant?	✗		✗		✗		Facebook ✓		✗		Google+ ✓	
		N. America hot fetch comparison						**N. America cold fetch comparison**					
N. America	$c_{A,B,v}$ **frac.**	0.3731	0.2537	0.2388	0.3433	0.2090	0.5522	0.8507	0.0149	0.5375	0.2985	0.0448	0.8806
	P-value	5.0e−02	6.7e−05	2.2e−05	1.4e−02	1.8e−06	4.6e−01	4.0e−09	9.2e−19	6.3e−01	1.3e−03	6.8e−16	1.0e−10
	Significant?	✗		✗		✗		Facebook ✓		✗		Google+ ✓	
		Europe hot fetch comparison						**Europe cold fetch comparison**					
Europe	$c_{A,B,v}$ **frac.**	0.1842	0.4868	0.1315	0.6711	0.1053	0.5789	0.5657	0.1579	0.2368	0.5921	0.0263	0.9211
	P-value	2.3e−08	9.1e−01	3.0e−11	3.8e−03	5.6e−13	2.1e−01	3.0e−01	1.0e−09	4.7e−06	1.4e−01	7.7e−20	6.3e−15
	Significant?	✗		Google+ ✓		✗		✗		✗		Google+ ✓	

true nodes is significant. Since majority of the comparisons reveals statistically significant differences in performance between Flickr and other Photo CDNs, we can conclude with sufficient confidence that for our given dataset, Flickr's performance is indeed inferior.

Table 5 presents the complete results of all Photo CDNs. In addition to the DBit results, we show the comparison based on first order statistics (mean, median and 95th percentile) in Table 4. The first order statistics are obtained by aggregating the samples across all vantage points.

We use the results in Table 4 in conjunction with Tables 3 and 5 to demonstrate how ambiguities can arise due to comparisons based on first order statistics and how DBit is able to avoid them. For example, observe in Table 4, that with respect to cold fetches in Europe, Akamai performs the worst at 95th percentile, whereas Flickr performs the worst with respect to median and average. Though the difference at the 95th percentile is marginal, this shows an example of an instance where first order statistics do not clearly distinguish which Photo CDN is better. DBit on the other hand is able to provide a statistically significant verdict which declares Flickr to be slower than Akamai in Europe as shown in Table 3.

Similarly, notice that the comparison between Akamai and Facebook in Europe with respect to first order statistics is ambiguous (Table 4). While Akamai performs better with respect to the average and median, its performance is significantly worse at the 95th percentile. Deciding which Photo CDN performs better now

Table 6
Hot and cold fetch comparison for video CDNs .

Global hot and cold comparison for video				
	Hot fetch		Cold fetch	
	FB ∧ YT	YT ∧ FB	FB ∧ YT	YT ∧ FB
$c_{A,B,\nu}$ **frac.**	0.037	0.7778	0.2963	0.037
P-value	1.7e−32	6.0e−11	2.5e−06	1.7e−32
Significant	**YouTube** ✓		✗	

becomes a matter of judgement in this case. DBit's approach, on the other hand, provides a systematic framework to resolve this ambiguity, its comparison yields an insignificant result since it is unable to find statistically significant reasons to declare one better than the other (Table 5).

Finally notice that in Table 3 the hot fetch comparison between Flickr and Facebook is insignificant. Even though 59% of the nodes accept the hypothesis that Facebook is faster than Flickr, the final verdict is ruled to be insignificant. This happens because in Europe we have access to 76 vantage points, out of which 44 (59%) accept the hypothesis that Facebook is faster. However, because the total number of vantage points is small, DBit requires a greater share of vantage points (atleast 65%: 49 in this case, see Fig. 1(c)) to establish statistical significance, therefore resulting in an insignificant comparison. This points towards the ability of DBit to adjust its statistical significance threshold to the total number of vantage points available, hence only declaring significant performance differences when there is strong statistical basis to do so.

Video CDN comparison. For Video CDNs we compare the performance of Facebook with YouTube. Not much is known about YouTube's CDN, however, it is reasonable to expect that YouTube servers are co-located with other Google servers and hence YouTube CDN should be of similar scale. We confirm through probing that Facebook uses the same sites to serve videos it uses for its photo content.

We apply DBit to the Video CDN data. The hypothesis and the converse hypothesis we frame at every vantage point are that Facebook is faster than YouTube and YouTube is faster than Facebook, respectively. Table 6 shows the results for both hot and cold fetch comparison for videos. With respect to hot fetches we see that ∼ 78% of the vantage points indicate that hot fetches for YouTube are faster. This forms a significant fraction of vantage points and the probability of achieving this number on the basis of mere chance is extremely low, as shown by the value in the corresponding P-value row. Hence we conclude that for hot fetches YouTube performs faster.

On the other hand, we see that for cold fetches, for both the hypothesis the fractions of $c_{A,B,\nu}$ is low. This implies that majority of the vantage points did not see statistically significant distributional differences between Facebook and YouTube. Therefore, we do not have sufficient basis to conclude that statistically significant performance differences exist between the two and we mark the comparison result to be insignificant.

Together these results show how different CDNs can be compared using DBit. Specifically, they show what are the conditions under which we conclude statistically significant performance difference and conditions under which the performance differences are not significant.

5.3. What is the magnitude of differences?

In this section, we try to understand the magnitude of differences based upon which DBit detects statistical differences. To this end, we employ the following methodology for a quantitative evaluation. Say that DBit determines Flickr to be slower than Google+, then at the second stage of DBit we are able to select all those nodes where Flickr is slower than Google+. For these set of nodes, the measured latency will be higher for the Flickr nodes than the Google+ nodes on average. We calculate the magnitude of the differences between Flickr and Google+ fetch by subtracting each Google+ sample from Flickr sample, where both samples were acquired at the same time of day.[5] We then repeat the same procedure for a pair of CDNs where DBit does not identify statistical differences.

Fig. 3(a) shows the CDF of the differences between Flickr and other Photo CDNs for hot fetches. The difference between Flickr and other Photo CDNs at the medians is ∼ 80–95 ms and increases to ∼ 900 ms at the 90th percentiles. This shows that DBit is able to detect differences even when they exist at the scale of tens or hundreds of milliseconds. Fig. 3(b) shows the CDF of the absolute differences between Flickr and other Photo CDNs for cold fetches. The median differences for cold fetch are ∼ 600 ms for all Photo CDNs, likely large enough to be easily detected by DBit.

When DBit does not identify a statistical difference, what does the magnitude of the differences look like? Fig. 3(c) shows two CDFs one for the difference between Flickr and Facebook, the case where DBit indicates a statistical difference and another between Facebook and Akamai, the case where DBit does not detect statistical performance differences. Observe that the latter CDF lies above the former, implying that on average the magnitude of the differences for the Facebook–Akamai case is smaller. Moreover, the Flickr–Facebook case shows a larger proportion of nodes for which Flickr shows worse behavior than Facebook than vice versa, but the CDF for the Facebook–Akamai case is more symmetric.

This is better understood by looking at Fig. 3(d), which shows the scatter plot of all the Photo CDNs at the 95th percentile for each Planetlab node for Flickr's comparison with other Photo CDNs. Note that Flickr is clearly slower at many vantage points when compared to other Photo CDNs as evident from the spread in the plot. In contrast, Fig. 3(e) shows the scatter plot of the 95th percentile of latencies for Facebook and Akamai, the case where DBit does not identify performance difference. In this case, there is no clear performance separation between the two Photo CDNs. From these results we conclude that DBit is sensitive enough to detect differences where they exist and correctly hides differences where they do not exist or are not significant.

5.4. Is DBit general?

We now explore the range of questions which can be answered using DBit. Our goal here is to demonstrate the generality of DBit as a methodology for detecting statistical differences.

Constrained datasets. In this section we evaluate DBit's ability to work with constraind datasets to detect performance differences. Towards this end we use the data obtained from RIPE Atlas testbed. Recall that, on the RIPE Atlas, we are only able to obtain hot-fetch, because of platform limitations (Section 4). Moreover, on the RIPE Atlas platform, we are only able to use 1x1 pixel photos. The summarized version of RIPE results are shown in Table 7 which shows that DBit is indeed able to detect significant differences and the results are consistent with those obtained from the PlanetLab results (for Global, N. America and Europe). We refer the reader to [21] for RIPE results comparing all Photo CDNs.

Tail latency. Because latency impacts revenue, content providers are interested in engineering for tail latency [24]. DBit can also be used to study whether the tail latency distributions

[5] Say that a Google+ sample is denoted by x_{G,ν,t_n} and a Flickr sample is denoted by x_{F,ν,t_n} at node ν at time $t = n$. Then the difference between the sample is $x_{D,\nu,t_n} = x_{F,\nu,t_n} - x_{G,\nu,t_n}$. We can then obtain a single CDF of the differences over all the nodes in the $\mathcal{V}$.

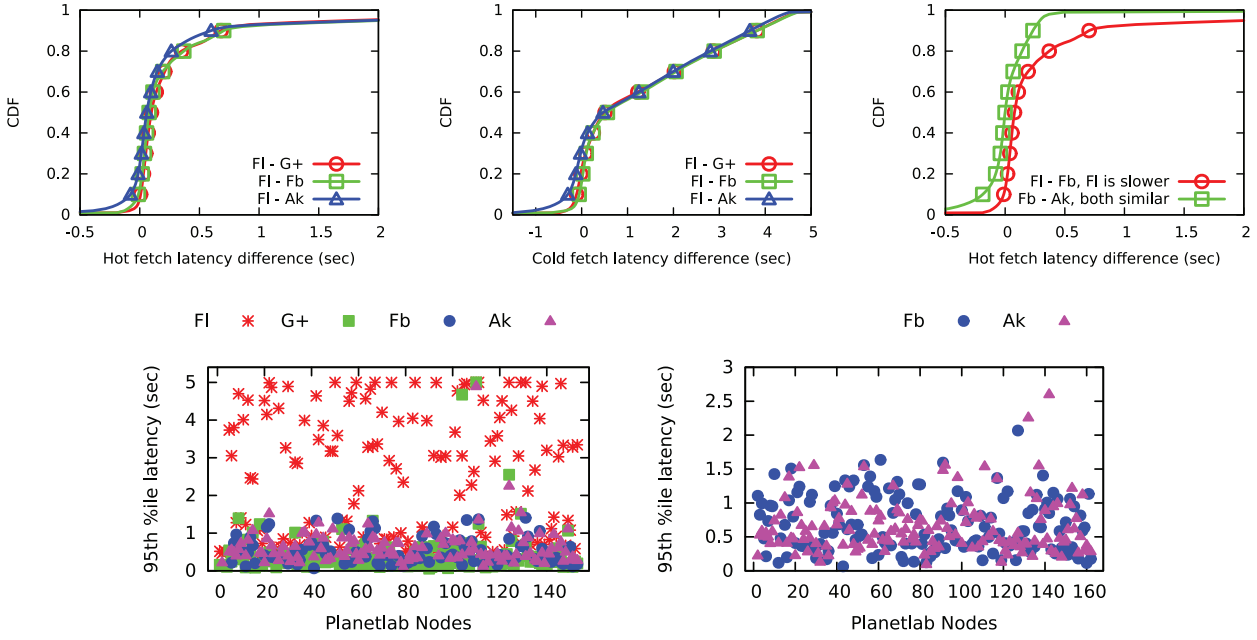

Fig. 3. (a) Difference CDF between Flickr and other Photo CDNs for hot fetches. (b) Difference CDFs between Flickr and other Photo CDNs for cold fetches. (c) CDFs where DBit identifies statistical differences and where it does not. (d) Scatter plot for hot fetches at 95th percentile where DBit detects statistical performance difference. (e) Scatter plot for hot fetches at 95th percentile where DBit does not detect difference.

Table 7
Results for hot fetch from RIPE Atlas probes using 1x1 pixel photo (size ~ 750 B). Latency here does not include DNS time.

	Global hot fetch comparison		
	Facebook vs. Flickr	Flickr vs. Akamai	Flickr vs. Google+
Global	Facebook ✓	Akamai ✓	Google+ ✓
Africa	✗	✗	Google+ ✓
Asia	✗	Akamai ✓	Google+ ✓
Europe	Facebook ✓	Akamai ✓	Google+ ✓
N. America	Facebook ✓	Akamai ✓	Google+ ✓
Oceania	Facebook ✓	✗	✗
S. America	Facebook ✓	Akamai ✓	Google+ ✓

of different CDNs differ significantly, by appropriately truncating the original distributions. For example we can obtain the data at the tail by only considering the samples at the 90th percentile and above. Table 8 shows the results for interesting performance differences that we find from our dataset. The results show that at the 90th percentile Google+ shows better performance compared to Facebook for both hot and cold fetches. Additionally, Google+ also performs better than Akamai for hot fetches at the 90th percentile. This suggests that while Facebook's optimized photo stack [5] allows comparable performance for the common case but Google's massive scale and full control over its serving infrastructure gives it advantage over Facebook and Akamai at the tail of the distribution.

Outliers. Using DBit, we can also detect outliers: "good" ("bad") clients which experience faster (slower) performance than all other clients. We find that most of the good clients are in North America or Europe. Two of our clients, located in Boston and Washington, are best across all Photo CDNs. We find other good performing clients in Canada and USA (4), Germany (3), France (2), Ireland and UK one each. We find the worst client in Cyprus, and we find bad clients in Spain (3), Portugal (3), Greece (2), Brazil (2), Ecuador (1) and Jordan (1).

6. Discussion

DBit is a first step towards comparing CDN performance in a systematic manner and its decisions are based on statistically significant performance difference across a set of vantage points. For this purpose we have used the well known KS-Test, however, KS-Test is known not to be very sensitive if the differences exist at the tails of the distribution. Towards this end it may be possible to extend other statistical tests such as the Kuiper Test for a two-sample one sided version which we leave for future work.

Further, we have designed DBit to be used on data which is either routinely collected by real systems or can be obtained without a need for sophisticated measurement methodologies. Such data sets are collected through number of approaches (1) regular server side measurements [24] (2) client-side injected measurements [9,25] or (3) emulated clients such as [11,26] and may include metrics such as latency and througput. Though we use

Table 8
Tail latency for both hot and cold photo fetches from the global set of Planetlab clients.

		Comparison at 90th percentile					
		Hot fetch				Cold fetch	
		Facebook vs. Google+		Akamai vs. Google+		Facebook vs. Google+	
		FB >G+	G+ >FB	AK >G+	G+ >AK	FB >G+	G+ >FB
	$c_{A,B,v}$ **frac.**	0.7037	0.216	0.784	0.1542	0.6975	0.2716
	P-value*	2.3e−07	1.0−e13	1.9e−13	6.6e−20	5.4e−07	5.3e−09
Global	**Significant?**	Google+✓		Google+✓		Google+✓	

latency as the metric to demonstrate how DBit can be used, DBit itself is general and can be used with any other performance metric. As future work, we plan to use DBit in determining CDN performance difference on data collected from real world clients. Towards this end, we plan to take approach 2 from above by implementing a browser plugin which can be installed by users opting in to the measurement study. This allows us to capture data which is representative of user performance in the wild and exposes opportunities to use DBit in understanding real world performance comparison. Additionally, it allows to instrument a range of web applications such as video streaming and search *etc.* Finally, in this paper we focus on applying DBit to compare performance for a single metric (i.e., latency), we leave it to future work to extend DBit to incorporate more than one metric.

7. Related work

Prior work on CDN performance analysis ranges from modelling to measurement studies for CDN comparison [4,13,20,22,27]. Our work is complementary to these and proposes a methodology to rigorously compare aspects of CDN performance. Other notable work attempts to understand peer-to-peer content delivery systems [8] and ISPs [7]. Their methodology compares totals, and first and second-order statistics across a variety of properties: content object sizes, bandwidth demands, transaction rates, path latency, and path stretch. By contrast, our work focuses on a different setting (photo sharing) and on one metric, latency, but explores distributional differences. Finally, Tariq et al. [28] propose a What-If scenario evaluator to understand how changes to network infrastructure will translate to user perceived performance. By contrast, DBit can help a CDN understand what aspects of its network infrastructure it needs to improve in order to match a competitor's performance.

8. Conclusion

In this paper, we propose a novel methodology to detect statistical differences in performance for Content Distribution Networks and large scale distributed systems in general. DBit is powerful, enabling not only comparisons across CDNs, but can also be used to answer a range of questions such as identifying differences in tail performance and regional performance. DBit is a first step towards a comprehensive methodology for CDN evaluation and much work remains: exploring other metrics such as throughput.

References

[1] Apple building its own CDN, 2012, URL: http://blog.streamingmedia.com/2014/02/apple-building-cdn-software-video-delivery.html.
[2] Business Insider: NetFlix building its own CDN, 2012, URL: http://goo.gl/PXi8yJ.
[3] Microsoft Azure Content Delivery Network, 2012, URL: http://blog.streamingmedia.com/2014/02/apple-building-cdn-software-video-delivery.html.
[4] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, R. Govindan, Mapping the Expansion of Google's Serving Infrastructure, in: Proceedings of the ACM Internet Measurement Conference (IMC '13), 2013.
[5] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, H.C. Li, An analysis of Facebook photo caching., in: SOSP, ACM, 2013, pp. 167–181.
[6] C. Huang, A. Wang, J. Li, K.W. Ross, Measuring and evaluating large-scale CDNs, Technical Report, MSR-TR-2008-106, Microsoft, 2008.
[7] R. Mahajan, M. Zhang, L. Poole, V.S. Pai, Uncovering performance differences among backbone isps with netdiff., in: NSDI, USENIX Association, 2008, pp. 205–218.
[8] S. Saroiu, K.P. Gummadi, R.J. Dunn, S.D. Gribble, H.M. Levy, An analysis of internet content delivery systems, 5th Symposium on Operating Systems Design and Implementation (OSDI) 2002, Boston, MA, USA, 2002.
[9] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, J. Padhye, Analyzing the performance of an Anycast CDN, in: Proceedings of the ACM Internet Measurement Conference (IMC'15), 2015.
[10] Conviva, Conviva, URL: http://www.conviva.com/.
[11] Keynote, KeyNote, 2012, URL: http://www.keynote.com/solutions/testing/load-testing.
[12] D. Beaver, S. Kumar, H.C. Li, J. Sobel, P. Vajgel, F. Inc, Finding a needle in haystack: Facebook'ss photo storage, in: Proceedings of OSDI, 2010.
[13] Y. Chen, S. Jain, V.K. Adhikari, Z.-L. Zhang, Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution., in: Internet Measurement Conference, ACM, 2011, pp. 559–568.
[14] E.W. Weisstein, Kolmogorov-Smirnov Test, 2013, URL: http://mathworld.wolfram.com/Kolmogorov-SmirnovTest.html.
[15] GraphPad Software - Binomial Test, 2013, URL: http://www.graphpad.com/guides/prism/6/statistics/index.htm?stat_binomial.htm.
[16] Beware of the KS-Test, 2013, URL: https://asaip.psu.edu/Articles/beware-the-kolmogorov-smirnov-test.
[17] Common mistakes in using statistics, 2013, URL: https://www.ma.utexas.edu/users/mks/statmistakes/StatisticsMistakes.html.
[18] D. Freedman, R. Pisani, R. Purves, Statistics, International student edition, W.W. Norton & Company, 2007. (Chapter 6).
[19] S. Banerjee, T.G. Griffin, M. Pias, The interdomain connectivity of PlanetLab nodes, Technical Report.
[20] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl, Globally distributed content delivery, In IEEE Internet Computing, 2002.
[21] Z. Akhtar, A. Hussain, E. Katz-Bassett, R. Govindan, DBit: a methodology for comparing content distribution networks, Technical Report, 15-957, University of Southern California, Computer Science Department, 2015.
[22] Y. Chen, S. Jain, V.K. Adhikari, Z.-L. Zhang, K. Xu, A first look at inter-data center traffic characteristics via yahoo! datasets., in: INFOCOM, IEEE, 2011, pp. 1620–1628.
[23] H.H. Liu, Y. Wang, Y.R. Yang, H. Wang, C. Tian, Optimizing cost and performance for content multihoming, in: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, in: SIGCOMM'12, ACM, New York, NY, USA, 2012, pp. 371–382, doi:10.1145/2342356.2342432.
[24] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, R. Govindan, Reducing web latency: the virtue of gentle aggression, in: Proceedings of the ACM SIGCOMM 2013, in: SIGCOMM '13, ACM, 2013, pp. 159–170, doi:10.1145/2486001.2486014.
[25] F. Chen, R.K. Sitaraman, M. Torres, End-user mapping: next generation request routing for content delivery, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, in: SIGCOMM '15, ACM, New York, NY, USA, 2015, pp. 167–181, doi:10.1145/2785956.2787500.
[26] A. Gupta, M. Calder, N. Feamster, M. Chetty, E. Calandro, E. Katz-Bassett, Peering at the internet's frontier: a first look at isp interconnectivity in africa, in: Passive and Active Measurement, Los Angeles, CA, 2014, pp. 204–213.
[27] P. Gill, M.F. Arlitt, Z. Li, A. Mahanti, Youtube traffic characterization: a view from the edge., in: Internet Measurement Comference, ACM, 2007, pp. 15–28.
[28] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, M. Ammar, Answering what-if deployment and configuration questions with wise, in: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, in: SIGCOMM'08, ACM, New York, NY, USA, 2008, pp. 99–110, doi:10.1145/1402958.1402971.

Zahaib Akhtar is a third year Ph.D. student at the University of Southern California Computer Science Department where he is co-advised by Ramesh Govindan and Alefiya Hussain. He obtained his M.S. and B.S. Computer Engineering degrees from Lahore University of Management Sciences (LUMS) SBASSE, Pakistan, where he was supervised by Ihsan Ayyub Qazi.

Alefiya Hussain is a Computer Scientist at the USC/Information Sciences Institute. Her research focuses on internet measurement, experimentation, and analysis of security attacks on networked systems. Her pioneering work on the measurement and characterization of denial of service attacks was published in top-tier conferences as well as referenced in graduate courses across the nation. From 2005 to 2010, Dr. Hussain was a Senior Principal Scientist at Sparta, where she developed learning mechanisms to optimize application performance over hostile and lossy networks. At USC, Dr. Hussain is leading a team to develop the experiment lifecycle management framework and enable complex and large scale cyber security experiments for DETERLab. She received a M.S. (2001) and Ph.D. (2005) in Computer Science from the University of Southern California and a B.E. (1997) in Computer Engineering from the University of Pune, India.

Ethan Katz-Bassett is an Assistant Professor in the Computer Science Department at the University of Southern California. His primary interests are in networks and distributed systems. His goal is to improve the reliability and performance of Internet services. To understand the problem space, he looks to the needs of operators and providers, and conducts detailed network measurements. Based on the insights gained from measurements, he designs deployable systems to improve the Internet and services that run over it and focuses on the two components needed for reliably fast Internet services: (1) the Internet must provide reliable, high performance routes for traffic, and (2) the need to architect quality services and protocols to use these routes, to take advantage of the Internet's strengths and mask its limitations.

Dr. Ramesh Govindan is the Northrop Grumman Chair in Engineering and Professor of Computer Science and Electrical Engineering at the University of Southern California (USC). Dr. Govindan received the B.Tech. degree from the Indian Institute of Technology at Madras, and the M.S. and Ph.D. degrees from the University of California at Berkeley. Prior to joining USC, he was a member of the technical staff at Bell Communications Research, a project leader at USC's Information Sciences Institute and at the International Computer Science Institute at Berkeley. Dr. Govindan's research has focused on scalable and robust routing infrastructures in large networks such as the Internet, on the structural properties of the Internet, and on the architectures and programming systems for wireless and mobile networks. He is a fellow of the ACM and of the IEEE, a former Editor-in-Chief of the IEEE Transactions on Mobile Computing, and a Distinguished Alumnus of the Indian Institute of Technology, Madras.