Content Retrieval using Cloud-based DNS

Ravish Khosla, Sonia Fahmy, Y. Charlie Hu
Purdue University
Email: {rkhosla, fahmy, ychu}@purdue.edu

***Abstract*—Cloud-computing systems are rapidly gaining momentum, providing flexible alternatives to many services. We study the Domain Name System (DNS) service, used to convert host names to IP addresses, which has historically been provided by a client's Internet Service Provider (ISP). With the advent of cloud-based DNS providers such as Google and OpenDNS, clients are increasingly using these DNS systems for URL and other name resolution.**

Performance degradation with cloud-based DNS has been reported, especially when accessing content hosted on highly distributed CDNs like Akamai. In this work, we investigate this problem in depth using Akamai as the content provider and Google DNS as the cloud-based DNS system. We demonstrate that the problem is rooted in the disparity between the number and location of servers of the two providers, and develop a new technique for geolocating data centers of cloud providers. Additionally, we explore the design space of methods for cloud-based DNS systems to be effective. Client-side, cloud-side, and hybrid approaches are presented and compared, with the goal of achieving the best client-perceived performance. Our work yields valuable insight into Akamai's DNS system, revealing previously unknown features.

I. Introduction

The Domain Name System (DNS) [13] – mostly used to convert names to IP addresses – is an integral service in the Internet. The name resolution service has been traditionally offered by Internet Service Providers, with servers close to the client [9] (referred to as *local DNS*). DNS is often used by Content Distribution Networks (CDNs) to redirect clients to the nearest data center [11], [17]. Hence, when the local DNS server queries CDNs such as Akamai to identify content servers, the CDNs return servers close to the local DNS, which in most cases is close enough to the client.

With the emerging trend of cloud computing, a host of services including DNS are being offered by the cloud, *e.g.* Google [6] and OpenDNS [15]. These cloud DNS services not only provide fast DNS resolution due to larger caches, but may also provide security benefits, protecting against DNS cache poisoning and Denial-of-Service (DoS) attacks [6]. However, there can be potentially high latencies between the client and the resolved servers, degrading client performance [1]. This effect is pronounced when obtaining servers for a highly distributed CDN such as Akamai. Huang *et al.* [9] estimate that the server latency increases by as much as 193 ms at the 95^{th} percentile when using cloud-based DNS systems, compared to local DNS. This is unacceptable, especially since Akamai's network is often used for streaming video.

Akamai is the dominant content provider, delivering between fifteen and thirty percent of all Web traffic, reaching more than 4 Terabits per second [2]. This makes the problem of remote Akamai content servers returned by using cloud-based DNS systems critical. In this paper, we investigate this problem with a case study of Akamai-hosted content as accessed by clients using Google DNS. We first geolocate the Google DNS and Akamai servers. One of the key challenges we face is that Google DNS uses IP anycast and hence the location of its servers hosted at Google data centers cannot be found using simple IP geolocation. We therefore develop a novel technique for geolocating Google data centers, and find that Google's DNS servers oftentimes do not see closeby Akamai servers. We also find that the Google DNS servers are placed more sparsely around the world than Akamai's servers, yielding poor client performance when accessing Akamai's content using Google DNS.

We then present and compare alternative solutions to the problem. We posit that cooperation among cloud providers, those which host content and those which host DNS services, is the best solution. However, in the absence of such cooperation, we design a hybrid client-cloud approach which queries specific Akamai name servers whose IP address has been found using cloud DNS. We find that the servers returned by this hybrid approach are usually the same as those returned by local DNS, preserving the performance advantage of local DNS. Our results also shed light onto Akamai's network, demonstrating that Akamai's DNS servers do respond to queries even when asked out of turn, albeit after a potential delay.

The contributions of our paper include:

- We present a novel, lightweight geolocation technique for locating cloud data centers (Section II-B).
- We use our geolocation technique to gain insight into the problem of poor client performance in accessing content through cloud-based DNS (Section IV).
- We propose and compare solutions to this problem (Section V). We also present a hybrid client-cloud approach that a client can use in today's Internet.

The rest of the paper is organized as follows. Section II provides an overview of DNS systems of Akamai and Google. Section III defines the problem while Section IV investigates the causes of this problem. We compare various solutions to the problem in Section V. We summarize related work in Section VI and conclude in Section VII.

II. Cloud-based DNS Systems

We now study DNS systems of two different kinds of clouds: Akamai's CDN and Google's DNS.

A. Akamai DNS Primer

Akamai uses two levels of DNS servers to redirect clients to the closest content server [17]. We use an example of an iterative DNS query to illustrate the steps involved (Figure 1). Suppose a client queries its local DNS for *videos.buy.com*. Either the local DNS knows the answer from its cache, or it queries top level and Akamai DNS servers and returns the canonical name (CNAME) *videos.buy.com.edgesuite.net*. The client then queries the local DNS for this CNAME and receives another CNAME *a1507.b.akamai.net*. We now use the command *dig +trace* [5] from the client to follow name server referrals during resolution, while eliminating caching. The client queries the top level domain server *j.root-servers.net* for *a1507.b.akamai.net*, which returns a list of name servers out of which the client chooses *c.gtld-servers.net* and queries it, which gives a list of Akamai's top level name servers. The client chooses *zh.akamaitech.net* for query in the next step, which returns Akamai second level name servers whose IP address is dependent upon the client's location (i.e., proximity-aware). Overall, there are nine second level name servers for this CNAME, from *n0b.akamai.net* to *n8b.akamai.net*. The client then chooses *n3b.akamai.net*, querying it for *a1507.b.akamai.net* and obtains the content server 149.165.180.19[1].

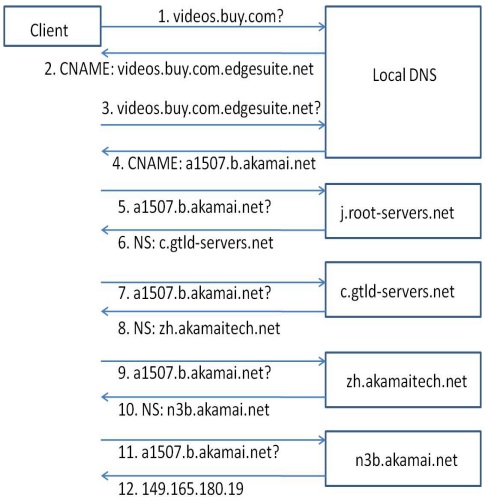


Fig. 1. Steps taken by a client in obtaining content server for an Akamai-hosted web site

In our experiments, we start with known Akamai CNAMEs like *a1507.b.akamai.net* and observe whether changing the number (1507) or the letter (b) gives us a CNAME which resolves to an Akamai content server. The number corresponds to a channel [23], whereas the letter corresponds to the way channels are grouped. Using the above technique, we discover eleven Akamai CNAME categories, listed in Table I with their respective name servers. We find that for each of the categories, channel numbers 0 to 4094 lead to valid CNAMEs, which map to edge servers IPs within the same Class C subnet or /24 prefix. Since there are at most 256 IPs in a Class C subnet, the average number of channels mapping to an edge server is about 16, possibly for load balancing purposes [23].

TABLE I
AKAMAI CNAMES STUDIED IN THIS PAPER WITH THEIR RESPECTIVE NAMESERVERS

CNAME category	Nameservers
x = 0 to 4094 , y = 0 to 8 for all rows unless specified otherwise	
a{x}.b.akamai.net	n{y}b.akamai.net
a{x}.c.akamai.net	n{y}c.akamai.net
a{x}.f.akamai.net	n{y}f.akamai.net
a{x}.h.akamai.net	n{y}h.akamai.net
a{x}.k.akamai.net	n{y}k.akamai.net
a{x}.l.akamai.net	n{y}l.akamai.net
a{x}.p.akamai.net	n{y}p.akamai.net
a{x}.vmg0.akastream.net	n{y}vmg0.akastream.net y = 0 to 6
a{x}.vmg2.akastream.net	n{y}vmg2.akastream.net y = 0 to 6
a{x}.uqg0.kamai.net	n{y}uqg0.kamai.net y = 0 to 6
a{x}.gi3.akamai.net	n{y}gi3.akamai.net

B. Geolocating Servers in the Cloud

Extensive research exists on geolocating IP addresses in the Internet [14] (a detailed discussion of which is outside the scope of this paper). In this paper, we use the commercial geolocation tool GeoIP City provided by MaxMind [12] to geolocate IP addresses, which is accurate up to 25 miles. Using this service, we can easily geolocate Akamai content servers and nameservers with reasonable accuracy. For example, in Figure 1, we geolocate the end-server 149.165.180.19 to Bloomington, Indiana, which is found to be 85 miles away from our client IP with a Geo-RTT [10] of 1 ms, which matches the measured RTT of 1.5 ms.

However, Google DNS [6] uses IP anycast and both of its DNS IP addresses resolve to Mountain View, California. This demonstrates the difficulty of geolocating Google's data centers, which host Google DNS servers [4]. One of the solutions to this problem is presented in [9], which requires an infrastructure setup and is passive, waiting for clients to visit a popular web site. In contrast, we design a novel lightweight *active* technique for geolocating Google data centers. We run 1000 traceroutes (running for 12 hours) to the Google Public DNS IP 8.8.8.8 from 575 PlanetLab [19] nodes. We define VGDNS, which is the *Virtual Google DNS IP*, as the last hop right before the Google DNS IP in the traceroutes. We verify that these IPs indeed belong to Google using BGP routing tables from RouteViews [24]. We collect all such VGDNS IPs across the traceroutes from PlanetLab nodes and obtain 1477 unique IP addresses, with 46 unique locations. To geolocate Google data centers, we use hierarchical clustering algorithms [26] to cluster the 46 unique VGFE locations using Matlab. We compute the distance between two locations using Haversine Formula [22]. and cluster them using the

[1]While Akamai usually returns two content servers for each query, we use the first one in this paper.

agglomerative complete link clustering technique [26], using 50 miles as the cutoff distance between clusters. Since the accuracy of MaxMind is 25 miles, two IPs at the same location can be no more than 50 miles apart. This gives *40 clusters* out of the 46 unique locations. In the absence of ground truth, this number cannot be validated. However, it is sufficient for explaining the cloud interactions in this paper (Section IV).

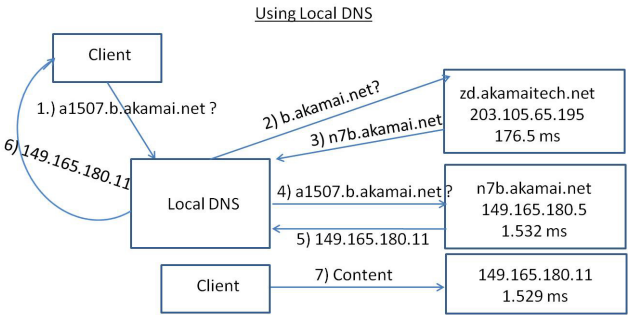
For locating Akamai data centers, we geolocate the content servers obtained by PlanetLab clients, as they resolve 11 random Akamai CNAMEs (one each from each row of Table I) through local as well as cloud-based DNS (1000 iterations each). We obtain 3223 unique IP addresses, which geolocate to 260 unique locations and *123 clusters*. We point out that two identical experiments uncover about *three times* as many Akamai data centers as Google, indicating more extensive presence of Akamai, compared to Google.

III. The Problem

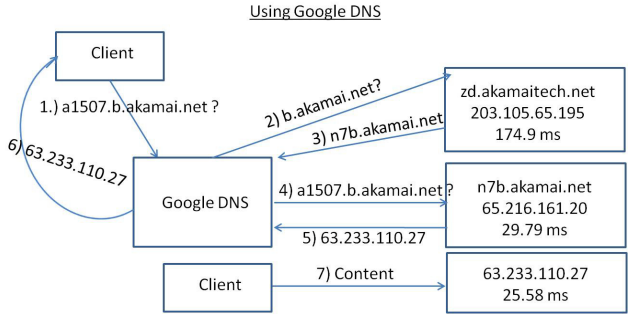
The problem we are investigating in this work is the high latency to the Akamai content servers that a client is redirected to when using cloud-based DNS systems. Figure 2 illustrates an example of the problem. We use the CNAME *a1507.b.akamai.net* (Section II-A), and resolve it using local DNS and Google DNS. We choose a case where both resolutions seem to proceed exactly the same as far as the DNS server names are concerned. However, as Figure 2 shows, the actual server IP addresses and their latencies from the client are different, with the Google DNS suffering because Akamai returns the IP addresses of the name server and content server which are close to the Google data center. This problem has been documented in [1], [9].

We now quantitatively demonstrate the existence of the high latency Akamai servers to the client when cloud-based DNS is used. As mentioned in Section II-A, each of the 4095 CNAMEs in a category of Table I map to 256 content servers within the same /24 prefix. We randomly select n CNAMEs such that we expect to see all 256 edge servers, with n to be determined. This problem is equivalent to ball-selection problem, which has been solved in [21] and, using their result in our context, we find n = 1568. Adding in the cases with known CNAMEs, (e.g. *a1507.b.akamai.net* for *videos.buy.com*), we obtain 1571 CNAMEs per category of Table I, which we use for all experiments below.

We probe the CNAMEs using the local DNS of each of the 575 PlanetLab nodes and then using Google DNS. We measure the quality of servers returned by pinging the servers with three ICMP echo request packets and noting the minimum RTT, which reduces RTT inflation due to network congestion to a certain extent. We use this technique for latency measurement throughout this paper. For each CNAME category, we compute the mean difference in latency between the client and the server resolved through cloud-based DNS and local DNS, considering the different server cases only. This mean latency inflation is averaged across all CNAME categories and then across all nodes. Our results show that the average latency inflation is 14.15 ms for Google DNS, which is 720.5% in



(a) Resolution through local DNS, indicating IPs and the RTTs from client



(b) Resolution through Google DNS, indicating IPs and the RTTs from client

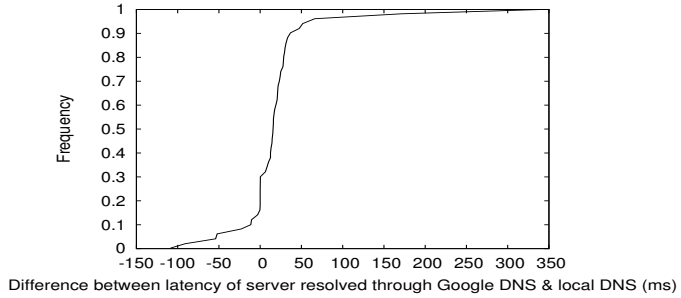
Fig. 2. Comparison of DNS lookup of *a1507.b.akamai.net* through local DNS and Google Public DNS

percentage terms. While the absolute latency inflation numbers do not seem extremely large, they are significant for video streaming and dynamic content applications.

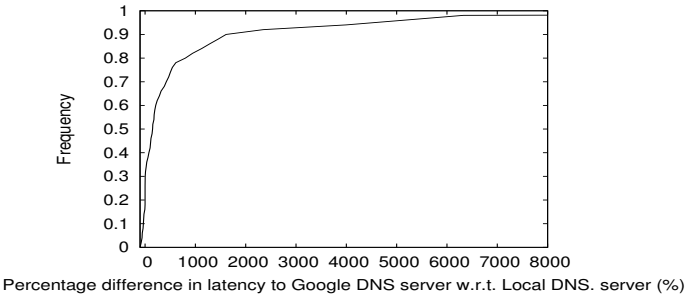
We plot the CDF of latency and percentage latency inflation for a typical CNAME in Figure 3. The CDF is computed with one data point per PlanetLab node. There are a few cases for which the inflation is negative. However, such cases are infrequent and are likely caused by large distances between the client and local DNS [9]. The results also show that the latency inflation has a heavy tail. While the *average inflation is around 15 ms*, around 17% of the clients experience inflation of more than 1000%.

IV. Causes

To identify the causes of latency inflation, for 1000 iterations run from PlanetLab nodes, we record the node IP C, VGDNS IP G, and the Akamai server IP corresponding to CNAME *a1507.b.akamai.net*, obtained through local DNS (server A) and through Google DNS (server A'). We then geolocate these four IP addresses and compute the geographical distance between the client C and the Akamai server it is redirected to A, g_{C-A}. We also compute the distance between the VGDNS IP G and the Akamai server it is redirected to A',

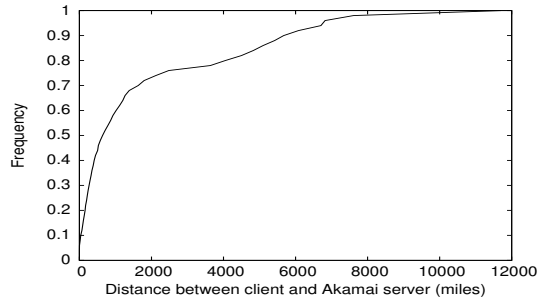


(a) CDF of latency inflation when using Google DNS as observed by a client

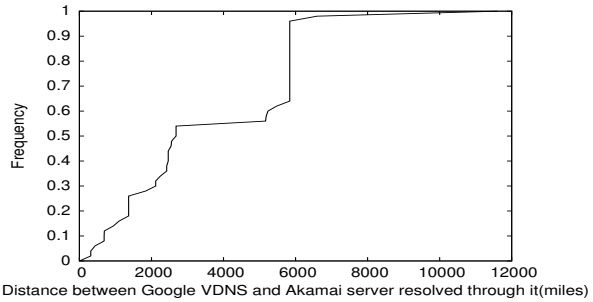


(b) CDF of percentage latency inflation when using Google DNS as observed by a client

Fig. 3. Quantifying performance degradation using cloud-based DNS w.r.t. local DNS for CNAME *a{x}.c.akamai.net*



(a) CDF of g_{C-A}, the geographical distance between Client and Akamai server resolved through local DNS



(b) CDF of $g_{G-A'}$, the geographical distance between VGDNS and Akamai server resolved through Google DNS

Fig. 4. Comparing distances of Akamai content servers from the resolution node for client and Google DNS

$g_{G-A'}$. The results are combined across iterations and across nodes to obtain median g_{C-A} as 643 miles. The median $g_{G-A'}$ is 2683 miles, which is substantially higher than g_{C-A}. The CDF of these two distances is shown in Figure 4. We observe jumps at discrete distances in Figure 4(b), because of the small number of data center locations, which will cause some iterations to be grouped together. The plots show that Google DNS sees an Akamai server which is much farther away from it than a client seeing a corresponding Akamai server.

We also compute, for each iteration, the percentage difference of $g_{G-A'}$ w.r.t. g_{C-A} and find the median to be 101%, which implies that $g_{G-A'}$ *is twice as much as* g_{C-A} in the median case. This result is interesting assuming Akamai does not discriminate among clients. This implies that even if the client was colocated with the Google DNS server, it would still attain lower performance than an average Internet client. We contend that this is due to two reasons. First, Google performs prefetching of name resolutions [6], which does not work well for Akamai-hosted dynamic content, which changes name resolutions in a matter of seconds [17]. Second, Google as a cloud is spread out over significant distances and may share its DNS resolutions among its data centers. As a result, it may not necessarily query Akamai's server from the DNS server which resolves client requests.

In our experiments, we compute the median g_{C-G}, which is the distance between the client and the VGDNS IP address. to be *5374 miles*. We also compute the percentage difference of g_{C-G} w.r.t. g_{C-A} for each iteration and find this to be 88% in the median case, showing that *Akamai servers are usually located closer to the client than Google DNS servers*. This further indicates that Google's DNS presence is sparse in the world, as shown by results of Section II-B and [9]. Coupled with the sub-optimal Akamai servers seen by Google nodes, this leads to significantly poorer performance of clients in accessing Akamai content through Google DNS.

V. Solutions

We now explore the solution space of how a client can best use cloud-based DNS to access content hosted by Akamai. We summarize the solutions in Table II.

A. Changes to DNS

A possible solution is based on a proposal initiated by Google researchers (see IETF draft [3]). This proposal requires changes to the DNS requests and replies by allowing recursive DNS resolvers to expose a portion of the client IP address to Akamai's CDN network, which it *may* use for returning

TABLE II
SOLUTIONS FOR OBTAINING GOOD CLIENT PERFORMANCE WHEN ACCESSING AKAMAI-LIKE CONTENT USING CLOUD-BASED DNS

Solution	Pros	Cons
Changes to DNS by revealing client IP to Akamai thereby enabling it to determine its closest server to the client	Correct Solution	The need for complete deployment across the Internet and ensuring backward compatibility with existing DNS
Increasing DNS data centers	Some performance improvement expected	Infrastructure spending and no guarantee of improved performance
Cooperation among clouds	Best solution with varying degrees of cooperation possible	Agreements and trust setup
Hybrid client-cloud approach	Good resolved server performance	Requires client to potentially wait for resolution. The technique based on reverse-engineering Akamai is temporary as it depends on Akamai implementation.

a client-optimized server. The primary drawback of this approach is that it requires changes to the DNS protocol which may not be universally adopted.

B. Cooperation among Clouds

We posit that the best solution is to have cloud-based DNS providers such as Google cooperate with CDNs like Akamai, similar to AS peering. Various degrees of cooperation are possible, from where Google will have the responsibility of selecting an Akamai replica (similar to DONAR [16]) to where Google DNS forwards client requests to Akamai servers (similar to [3]). The primary drawback of this technique is that it requires agreements and trust between cloud providers, which may be difficult to establish in the real world.

C. Increasing DNS Data Centers

Yet another solution can be for cloud-based DNS providers such as Google to employ many satellite data centers [25]. This implies that anycast routing will redirect a client to a closer DNS server which perhaps will see an Akamai server close enough to the client. However, this solution involves a significant investment from DNS providers. Moreover, this does not solve the issue of Google seeing farther Akamai servers than a normal client due to prefetching (Section IV).

D. Hybrid Approach

The solutions presented above are not deployed in today's Internet. Hence, we present a *hybrid client-cloud approach* that a client can use to identify low-latency Akamai content servers while preserving the security and outsourcing benefits of cloud-based DNS.

In the hybrid approach, the client queries the Akamai second-level name server directly, which will cause a closeby content server to be returned. The client obtains the IP address of the appropriate Akamai name server using cloud-based DNS. Figure 5 shows the same example as Figure 2 but using this hybrid approach. The client queries Google DNS for obtaining the IP address of *n7b.akamai.net*, which it then queries for the CNAME obtaining the content server, the *same* as that returned by local DNS in Figure 2(a). This is a hybrid solution because it involves the use of cloud DNS to resolve the name server IP and a local solution to query the IP directly to obtain content servers. This solution can be implemented as a patch for the client-side DNS software. Its only overhead is unexpected but infrequent DNS queries to Akamai nameservers, which should be tolerable given improved client performance while accessing Akamai content.

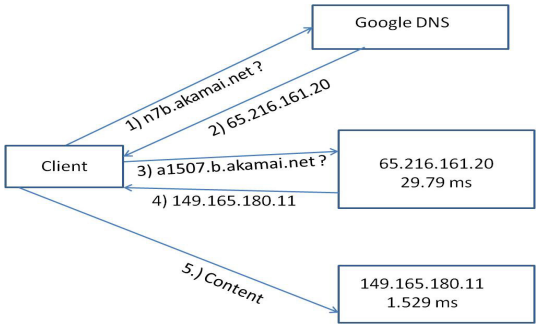


Fig. 5. Example of a hybrid approach for looking up Akamai content servers using Google DNS, showing IPs and the RTTs from client

A key aspect of this solution is that the client needs to know the name of the Akamai second-level name server, e.g., *n7b.akamai.net*. This can be built into the client-side DNS software, since Akamai uses predictable name server names. For example, a CNAME of $a\{x\}.\{z\}.akamai.net$ will have the name server $n\{y\}\{z\}.akamai.net$ with y ranging from 0 to 6 (Table I). A name server with any value of y will work and one can even choose y randomly for load balancing purposes. An alternate way to find the name of the name server is through the authority section of a *dig* [5], or to do a *dig +trace* for the CNAME using cloud-based DNS as the default DNS (assuming the client wishes to take advantage of its security features). This reveals the name of the name server.

Our results indicate that querying an Akamai name server, provided by cloud-based DNS, may or may not return a content server IP address. In case it does not, it returns a CNAME like *a1.b.akamai.net.0.1.cn.akamaitech.net*. However, if the client retries the query after some time, it is usually successful and receives an IP address which is the *same* as the one it would have received had it queried using local DNS. This indicates an Akamai content server returned to a client is *independent* of the Akamai name server queried. This is what makes this hybrid approach successful. We also find that there may be a slight delay before an arbitrary Akamai name

server resolves a CNAME. This delay is most likely due to background information sharing among various Akamai name servers, presumably with those close to the client. We find the typical delay to be less than 15 seconds (which was our retry period), except for a particular CNAME where the resolution does not succeed. A few seconds delay is an acceptable setup penalty for a typical long-lived Akamai session.

We conduct a measurement study similar to Section IV to investigate the effectiveness of the hybrid approach. We find that the hybrid approach *reduces* the median (mean) latency to a content server by around 7.5 ms (12.7 ms) as compared to the server obtained through Google DNS. These numbers are within 1 ms of the actual latency inflation caused by using cloud-based DNS as opposed to local DNS (Section III).

We also find that the hybrid approach returns the same server as local DNS in 45.1% of the cases. This is expected since Akamai returns two content servers and we choose the first one as the content server returned, resulting in around a 50% match. When the servers are different, we find the latency difference between the servers returned by the hybrid technique and the local DNS is less than a hundredth of a millisecond. This shows that the hybrid approach returns essentially the same servers as the local DNS, avoiding latency inflation due to cloud DNS.

VI. Related Work

Ager *et al.* [1] compare cloud-based DNS systems. While they show that the content servers returned by cloud DNS can be in different ASes from the client, they do not investigate causes and solutions to the problem. Cloud-based DNS systems are studied from a data center perspective in [9], demonstrating non-optimal client redirection using cloud-based DNS. However, they do not study a deeply distributed CDN like Akamai which handles dynamic content. Several studies have investigated data center performance [10], [20]. The WhyHigh tool [10] diagnoses high latency to Google's data centers and finds causes related to inter-domain routing, however effective solutions are not proposed.

There has been significant work [16], [7], [8], [18] on Global Traffic Management (GTM). GTM techniques redirect a client to the closest data center; however, this only enables the client to reach the closest cloud-based DNS server, which is not enough to ensure that good quality content servers are returned to the client.

VII. Conclusions and Future Work

Cloud DNS systems suffer from poor performance when a client accesses dynamic content hosted on highly distributed CDNs such as Akamai. In this paper, we have analyzed the reasons for performance degradation a client sees when using cloud-based DNS such as Google DNS to access Akamai-hosted content. We geolocated Google data centers using a novel technique based on active measurements. Our results show that sparse placement of Google DNS servers along with prefetching are likely to blame for sub-optimal content servers returned by Google DNS. We discussed several solutions to this problem, and posited that cooperation among clouds is the best solution. However, since no such solution is deployed today, we presented a hybrid client-cloud approach which returns servers comparable to local DNS. Our work raises important questions about the future cloud-based Internet, specifically the cooperation among clouds and which services should be migrated into the cloud. As future work, we plan to simulate different solutions to gain a better understanding of their advantages and disadvantages.

References

[1] B. Ager, W. Muehlbauer, G. Smaragdakis, and S. Uhlig. Comparing DNS Resolvers in the Wild. In *IMC*, pages 15–21, November 2010.

[2] Akamai. Akamai Customer Stories. http://www.akamai.com/html/customers/index.html, Retrieved December 2011.

[3] C. Contavalli and W. van der Gaast and S. Leach and D. Rodden. Client IP information in DNS requests. IETF Internet Draft draft-vandergaastedns-client-ip-00.txt, Jan 2010.

[4] Data Center Knowledge. Google Data Center FAQ. http://www.datacenterknowledge.com/archives/2008/03/27/google-data-center-faq/, March 2008.

[5] die.net. dig(1) - Linux man page. http://linux.die.net/man/1/dig, Retrieved December 2011.

[6] Google. Google Public DNS. http://code.google.com/speed/public-dns/, Retrieved December 2011.

[7] J. S. Gwertzman and M. Seltzer. The case for geographical push-caching. In *HotOS V*, 1995.

[8] C. Huang, N. Holt, Y. A. Wang, A. Greenberg, J. Li, and K. W. Ross. A DNS reflection method for global traffic management. In *USENIX ATC*, 2010.

[9] C. Huang, D. A. Maltz, A. Greenberg, and J. Li. Public DNS system and global traffic management. In *INFOCOM*, 2011.

[10] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *IMC*, pages 190–201, 2009.

[11] T. Leighton. Improving performance on the Internet. *Communications of the ACM*, 6(6):20–29, October 2008.

[12] MaxMind. MaxMind GeoIP City Database. http://www.maxmind.com/app/city, July 2011.

[13] P. Mockapetris. Domain names - implementation and specification. RFC 1035, November 1987.

[14] J. A. Muir and P. C. V. Oorschot. Internet Geolocation: Evasion and Counterevasion. *ACM Comput. Surv.*, 42:4:1–4:23, December 2009.

[15] OpenDNS. OpenDNS Basic. http://www.opendns.com/solutions/household/, Retrieved April 2011.

[16] P. Wendell and J. W. Jiang and M. J. Freedman and J. Rexford. DONAR: decentralized server selection for cloud services. In *SIGCOMM*, pages 231–242, 2010.

[17] J. Pan, Y. T. Hou, and B. Li. An overview of DNS-based server selections in content distribution networks. *Computer Networks*, 43(6):695 – 711, 2003.

[18] C. Partridge, T. Mendez, and W. Milliken. RFC 1546. Host Anycasting Service. http://www.ietf.org/rfc/rfc1546.txt, November 1993.

[19] PlanetLab. PlanetLab. http://www.planet-lab.org/, Retrieved December 2011.

[20] M. Saxena, U. Sharan, and S. Fahmy. Analyzing Video Services in Web 2.0: A Global Perspective. In *NOSSDAV*, May 2008.

[21] Thomas M. Sellke. How Many IID Samples Does it Take to See all the Balls in a Box? *The Annals of Applied Probability*, 5(1):294–309, February 1995.

[22] R. W. Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2):159, 1984.

[23] A. Su and A. Kuzmanovic. Thinning Akamai. In *IMC*, pages 29–42. ACM, 2008.

[24] University of Oregon. Route Views Project. http://www.routeviews.org/.

[25] Y. A. Wang, C. Huang, J. Li, and K. W. Ross. Estimating the performance of hypothetical cloud service deployments: A measurement-based approach. In *INFOCOM*, 2011.

[26] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.