Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure

Keqiang He, Alexis Fisher, Liang Wang, Aaron Gember, Aditya Akella, Thomas Ristenpart
University of Wisconsin – Madison
{keqhe,afisher,liangw,agember,akella,rist}@cs.wisc.edu

ABSTRACT

An increasingly large fraction of Internet services are hosted on a cloud computing system such as Amazon EC2 or Windows Azure. But to date, no in-depth studies about cloud usage by Internet services has been performed. We provide a detailed measurement study to shed light on how modern web service deployments use the cloud and to identify ways in which cloud-using services might improve these deployments. Our results show that: 4% of the Alexa top million use EC2/Azure; there exist several common deployment patterns for cloud-using web service front ends; and services can significantly improve their wide-area performance and failure tolerance by making better use of existing regional diversity in EC2. Driving these analyses are several new datasets, including one with over 34 million DNS records for Alexa websites and a packet capture from a large university network.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Network**]: General; C.4 [**Performance of Systems**]: Metrics—*performance measures*

General Terms

Measurement

Keywords

Web Service, Cloud Computing, EC2, Azure, DNS, trace analysis

1. INTRODUCTION

Up until a few years ago, web services were hosted in heterogeneous server clusters or co-location centers that were widely distributed across different network providers and geographic regions. Today, web services are increasingly being deployed in infrastructure-as-a-service (IaaS) clouds such as Amazon EC2, Windows Azure, and Rackspace. Industry and the media claim that over 1% of Internet traffic goes to EC2 [31] and that outages in EC2 are reputed to hamper a huge variety of services [4, 6, 24, 35].

IMC'13, October 23–25, 2013, Barcelona, Spain.

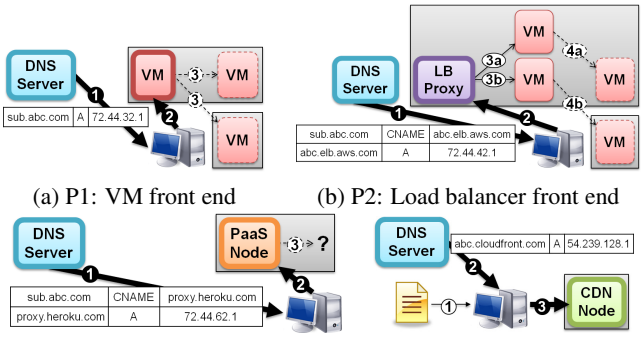
http://dx.doi.org/10.1145/2504730.2504740.

Despite the popularity of public IaaS clouds, we are unaware of any in-depth measurement study exploring the current usage patterns of these environments. Prior measurement studies have quantified the compute, storage, and network performance these clouds deliver [29, 30], evaluated the performance and usage patterns of specific services that are hosted in these clouds, e.g., Dropbox [23], or examined cloud usage solely in terms of traffic volume [28].

We present the first in-depth empirical study of modern IaaS clouds that examines *IaaS cloud usage patterns* and identifies *ways in which cloud tenants could better leverage IaaS clouds*. We focus specifically on web services hosted within IaaS clouds, which our study (unsurprisingly) indicates is a large and important use case for IaaS.

We first examine *who is using public IaaS clouds*. We generate a dataset of cloud-using domains using extensive DNS probing in order to compare the IPs associated with websites on Alexa's top 1 million list [1] against published lists of cloud IP ranges. This identifies that ≈40K popular domains (4% of the Alexa top million) have a subdomain running atop Amazon EC2 or Windows Azure, two of the largest public clouds. We extract an additional ≈13K cloud-using domains from a one week packet capture from a large university network, and we use this capture to characterize the network traffic patterns of cloud-hosted web services. These results indicate that a large fraction of important web services are already hosted within public IaaS clouds.

We proceed to dissect *how these services are using the cloud*. EC2 and Azure both have a veritable potpourri of features, including virtual machines, load balancers, platform-as-a-service (PaaS) environments, content-distribution networks (CDNs), and domain name services. They also give tenants the choice of deploying their services in several different regions (i.e., geographically distinct data centers), and EC2 provides several different "availability zones" within each region. We couple analysis of DNS records with two different cloud cartography techniques [34] to identify which features, regions and zones web services use. We identify several common front end deployment patterns and report estimates of the percentages of Alexa subdomains using each of the patterns. In particular, we find that about 4% of EC2-using web services use load balancers and 8% of them leverage PaaS. Only 5% of the DNS servers used by cloud-using subdomains run on VMs inside EC2 or Azure. We also show that 97% of the subdomains hosted on EC2 and 92% of the subdomains hosted on Azure are deployed in only a single region. Counted among these are the subdomains of most of the top 10 (by Alexa rank) cloud-using domains. Services deployed in EC2 also appear to make limited use of different availability zones: our measurements estimate that only 66% of subdomains use more than one zone and only 22% use more than two. This lack of redundancy means that many (even highly ranked



(a) P1: VM front end (b) P2: Load balancer front end

(c) P3: PaaS front end (d) P4: Leverage CDN

Figure 1: Deployment patterns for web services.

Alexa) services will not tolerate single-region or even single-zone failures.

Finally, we use a series of PlanetLab-based [17] active measurements and simulations to *estimate the impact of wide-area route outages and the potential for wide-area performance improvement*. We find that expanding a deployment from one region to three could yield 33% lower average latency for globally distributed clients, while also substantially reducing the risk of service downtime due to downstream Internet routing failures.

The remainder of our paper is organized as follows. We first provide background on Amazon EC2 and Windows Azure and discuss the primary datasets we use in our study (§2). We then examine who is using the cloud (§3), and which cloud features, regions, and zones are used by the cloud-using web services we identified (§4). Based on these observations, we proceed to estimate the wide-area-failure tolerance of current web service deployments and the potential for performance improvement (§5). Finally, we discuss related work (§6) before concluding (§7).

2. MEASUREMENT SCOPE & DATASETS

Public IaaS clouds, such as Amazon EC2, Windows Azure, and Rackspace, allow tenants to dynamically rent virtual machine (VM) instances with varying CPU, network, and storage capacity. Cloud tenants have the option of renting VMs in one or more geographically distinct data centers, or *regions*. Some clouds, such as EC2, further divide these regions into multiple distinct availability *zones*. Each zone has separate compute and power infrastructure to make certain failure modes zone-specific and to allow cloud tenants to replicate their deployments across multiple zones for smooth failover.

Beyond simple VMs, IaaS providers, as well as third parties, offer a wide-range of value-added features: load balancers (e.g., Amazon Elastic Load Balancer and Azure Traffic Manager), platform-as-a-service environments (e.g., Amazon Elastic Beanstalk, Heroku, and Azure Cloud Services), content-distribution networks (e.g., Amazon CloudFront and Azure Media Services), DNS hosting (e.g., Amazon route53), etc. The result is a complex ecosystem of interdependent systems operating at multiple layers of abstraction, and, in turn, a large variety of possible deployment patterns for cloud tenants. In this paper, we study four popular deployment patterns. We describe these using a series of examples.

In Figure 1, we show the steps involved in a client accessing an EC2-hosted web service that is using one or more of the aforementioned features. When a client wants to access a web service, it first performs a DNS lookup of the service's domain name. The response may contain an IP address associated with a VM (deployment pattern *P1*), a load balancer (*P2*), or a platform-as-a-service (PaaS) node (*P3*). With *P2*, the client request is subsequently directed to a VM[1]. Tenants using *P1–P3* may also rely on additional VMs or systems (dashed lines) to handle a client's request; these additional components may or may not be in the same region or availability zone (indicated by the gray boxes). An object returned to a client (e.g., a web page) may sometimes require the client to obtain additional objects (e.g., a video) from a content-distribution network (*P4*).

We focus on studying the front end portions of web service deployments within the above four deployment patterns (indicated by the thicker lines in Figure 1). These portions are encountered within the initial few steps of a client making a request. We leave an exploration of deployment/usage patterns covering the later steps (e.g. back-end processing) for future work.

2.1 Datasets

We use two primary datasets: (*i*) a list of cloud-using subdomains derived from Alexa's list of the top 1 million websites, and (*ii*) packet traces captured at the border of the UW-Madison campus network. Both datasets leverage the fact that EC2 [12] and Azure [8] publish a list of the public IPv4 address ranges associated with their IaaS cloud offerings. Below, we provide details on our Alexa subdomains and packet capture datasets. We augment these data sets with additional traces and active measurements to aid specific analyses; we describe these at the appropriate places in subsequent sections.

Top Cloud-Using Subdomains Dataset. Our first dataset is a list of subdomains which use EC2 or Azure and are associated with domains on Alexa's list of the top 1 million websites [1]. We consider a subdomain to use EC2 or Azure if a DNS record for that subdomain contains an IP address that falls within EC2 or Azure's public IP address ranges.

To construct this dataset, we first identified the subdomains associated with each domain on Alexa's list of the top 1 million websites. We started with Alexa's top 1 million list from Feburary 6, 2013 and attempted to issue a DNS zone transfer (i.e., a DNS query of type AXFR) for each domain on the list. The query was successful for only about 80K of the domains. For the remaining domains, we used dnsmap [16] to identify subdomains by brute-force. Dnsmap uses a pre-defined word list, which we augmented with the word list from knock [15], to construct potential subdomain names. Dnsmap then runs DNS queries to check if the potential subdomains actually exist. This brute-force approach misses some subdomains, but it allows us to provide a lower bound on the number of subdomains which use public IaaS clouds and explore the deployment patterns of these known cloud-using subdomains. We distributed this task to 150 globally-distributed PlanetLab nodes, producing a list of 34 million valid subdomains.

To limit the list of subdomains to cloud-using subdomains, we performed a series of DNS lookups using the UNIX dig utility. We first performed a single DNS lookup from one PlanetLab node (chosen from our set of 150 nodes) for each subdomain. If the DNS record contained an IP address within EC2 or Azure's public IP ranges[2], we included it on our list of the top cloud-using subdomains. This resulted in a list of 713K cloud-using subdomains. We then performed a DNS lookup for each of the cloud-using subdomains on every node in a set of 200 globally-distributed PlanetLab nodes. Figure 2 shows the geographic location of these PlanetLab

[1]Or PaaS nodes, as is done by Amazon Elastic Beanstalk and Azure Traffic Manager.

[2]We assume the IP address ranges published by EC2 and Azure are relatively complete.

Figure 2: PlanetLab nodes used for DNS lookups

nodes, which are spread across North America, South America, Europe, Asia, and Australia. The queries were performed March 27-29, 2013. These distributed DNS queries help ensure that we gather a comprehensive set of DNS records for each cloud-using subdomain and capture any geo-location-specific cloud usage.

We refer to the list of cloud-using subdomains, and their associated DNS records, as the Alexa subdomains dataset.

Packet Capture Dataset. Our second primary dataset is a series of packet traces captured at the border of the University of Wisconsin-Madison campus network[3]. We captured full IP packets whose source or destination IP address fell within the public address ranges published by EC2 and Azure. The capture was performed from Tuesday, June 26 to Monday, July 2, 2012 giving us a full week of traffic and a total of 1.4TB of data. The total Internet traffic averaged approximately 7Gbps during the capture, with about 1% of the traffic going to/coming from EC2 or Azure. Due to the relatively low rate of traffic being captured, no loss occurred during the capture process (according to tcpdump and counters reported by the border router). To protect user privacy, we anonymized the IP addresses of clients within the university network, and we only report aggregate statistics.

Since our traces contain full packets, we were able to perform an in-depth analysis of network and transport layer information (e.g., IP addresses, protocols, ports), application layer information (e.g., HTTP hostnames, HTTP content-type, HTTPS certificates), and packet payloads. We extracted relevant information from the traces using Bro [33], a network monitoring and traffic analysis tool. We refer to these traces as the packet capture dataset.

We recognize that a packet trace from a single campus vantage point may not reflect the "typically" usage patterns of services deployed in IaaS clouds. Correspondingly, we only leverage the packet capture for analysis which cannot be conducted using our Alexa subdomains dataset—namely, protocol usage (§3.1), popularity estimates based on traffic volume and flow counts (§3.2), and flow characteristics (§3.3).

3. WEB-FACING CLOUD TENANTS

In this section, we explore what applications are being hosted on public IaaS clouds. We start by analyzing the packet capture to identify the types of applications being hosted. This analysis suggests (unsurprisingly) that web applications represent a large, important set of cloud tenants. We then turn to examining which of the most popular websites are using clouds. We view popularity both globally, via the Alexa top website rankings, and locally, via the volume of traffic associated with each domain in the packet capture. We also analyze the traffic patterns of cloud-using services, including flow characteristics and content types served.

[3]The university has seven /24 IP blocks and one /16 IP block

Cloud	Bytes	Flows
EC2	81.73	80.70
Azure	18.27	19.30
Total	100	100

Table 1: Percent of traffic volume and percent of flows associated with each cloud in the packet capture.

	EC2		Azure		Overall	
Protocol	Bytes	Flows	Bytes	Flows	Bytes	Flows
ICMP	0.01	0.03	0.01	0.18	0.01	0.06
HTTP (TCP)	16.26	70.45	59.97	65.41	24.24	69.48
HTTPS (TCP)	80.90	6.52	37.20	6.92	72.94	6.60
DNS (UDP)	0.11	10.33	0.10	11.59	0.11	10.58
Other (TCP)	2.40	0.40	2.41	1.10	2.40	0.60
Other (UDP)	0.28	0.19	0.31	14.77	0.28	3.00
Total	100	100	100	100	100	100

Table 2: Percent of traffic volume and percent of flows associated with each protocol in the packet capture.

3.1 Protocols and Services

We first examine the fraction of bytes and flows in the packet capture that are associated with each cloud (Table 1). We only consider flows that were initiated within the university and destined for EC2 or Azure. We observe that the majority of cloud traffic, both as measured by volume and number of flows, is EC2-related: 81.73% of bytes (80.70% of flows) are associated with EC2, while Azure accounts for 18.27% of bytes (19.30% of flows).

Next, we use the packet capture to study the application-layer protocols used by cloud tenants. Table 2 shows the percentage of bytes (and flows) using a specific protocol relative to the total number of bytes (and flows) for EC2, Azure, and the capture as a whole.

We observe that more than 99% of bytes in the packet capture are sent and received using TCP, with less than 1% of bytes associated with UDP or ICMP. The vast majority of this TCP traffic is HTTP and HTTPS. The proportion of HTTPS traffic is far higher than that seen for general web services in the past (roughly 6% [18]); as we will show later, HTTPS traffic is dominated by cloud storage services. Interestingly, the majority of Azure's TCP traffic is HTTP (59.97%) while the majority of EC2's TCP traffic is HTTPS (80.90%)

The breakdown by flow count is less skewed towards TCP, with UDP flows accounting for 14% of flows in the packet capture. This is largely due to DNS queries, which account for 11% of flows but carry few bytes.

As one would expect, public IaaS clouds are also used for non-web-based services. In the packet capture, we find a small fraction of non-HTTP(S) TCP traffic and non-DNS UDP traffic going to both EC2 and Azure. This traffic includes SMTP, FTP, IPv6-in-IPv4, SSH, IRC, and other traffic that Bro could not classify.

Summary and implications. While we analyze a single vantage point, our measurements suggest that web services using HTTP(S) represent an important set of WAN-intensive cloud tenants. The extent to which compute-intensive workloads (that may not result in a large impact on network traffic) are prevalent as cloud tenants remains an interesting open question. In the following sections we dig into what tenants are hosting web services on public clouds as well as diving deeper into their traffic patterns.

3.2 Popular Cloud-Using (Sub)Domains

Cloud-using Alexa domains. We now consider what subset of the Alexa top 1 million websites use the cloud to (partly) host their services. Recall that Alexa provides an estimate of the most pop-

Provider	# Domains (%)	# Subdomains (%)
EC2 only	3,277 (8.1%)	685,725 (96.1%)
EC2 + Other	34,721 (86.1%)	21,628 (3.0%)
Azure only	184 (0.5%)	6,328 (0.9%)
Azure + Other	1,858 (4.6%)	225 (<0.01%)
EC2 + Azure	293 (0.7%)	4 (<0.01%)
Total	40,333 (100.0%)	713,910 (100.0%)
EC2 total	38,291 (94.9%)	707,357 (99.1%)
Azure total	2,335 (5.8%)	6,557 (0.9%)

Table 3: Breakdown of domains and subdomains based on their use of EC2, Azure, and/or other hosting services.

ular domains worldwide. Their ranking is based on the number of unique visitors and the number of page views over the last 3 months, aggregated at the domain level[4] [1]. Using our Alexa subdomains dataset, we can determine which Alexa sites are hosted in EC2/Azure.

We find that 40,333 (>4%) of the domains on Alexa's top 1 million list have a subdomain that uses EC2 and/or Azure. Under these domains, there are a total of 713,910 cloud-using subdomains. Note that these are lower bounds on cloud use, since our analysis approach (see §2.1) means we do not flag as cloud-using any domains that use a layer of indirection (e.g., via services like CloudFlare [11]) before requests are sent to EC2 or Azure.

Table 3 provides a breakdown of the domains and subdomains in terms of whether they use EC2, Azure, or other hosting services (the last indicating IP addresses not associated with EC2 or Azure). Note that "other" could in fact be public clouds besides EC2 and Azure. A subdomain is marked as *EC2 only* if it always resolves only to IP addresses within EC2; similarly for Azure. We mark a subdomain as *EC2+Azure*, *EC2+Other*, or *Azure+Other* if it resolves to IP addresses associated with the appropriate subset of EC2, Azure, and other. Domains are counted as *EC2 only* if all of their subdomains only use EC2; similarly for Azure. Domains are marked as *EC2+Azure*, *EC2+Other*, or *Azure+Other* if they have subdomains associated with the indicated subset of EC2, Azure, and other.

The vast majority of cloud-using domains (94.9%) use EC2, and the majority of these domains use other hosting for some of their subdomains (i.e., EC2 + Other). Only 5.8% of domains use Azure. Additionally, a small fraction (0.7%) of cloud-using domains use both EC2 and Azure; hence the *EC2 total* and *Azure total* rows in Table 3 sum to more than 100%. A list of the top 10 (by Alexa rank) EC2-using domains appears in Table 4. This list will be used in several later sections with results specific to EC2, which is why we excluded the four top Azure domains that would otherwise have been in the top 10: live.com, msn.com, bing.com, and microsoft.com.

The distribution of Alexa ranks for cloud-using domains is skewed: higher ranked domains are more likely to be cloud-using than lower ranked domains. Notably, 42.3% of cloud-using domains have ranks in the first 250,000 sites versus only 16.2% of the bottom 250K domains.

The most frequent prefix used by cloud-using subdomains in our Alexa subdomains dataset is www (3.3% of all cloud-using subdomains). The other top 10 prefixes (each <1%) are, in order: m, ftp, cdn, mail, staging, blog, support, test, and dev. The majority of subdomains are hosted either only in the cloud or only elsewhere, although a small fraction (3%) appear to be hosted both on EC2 and other providers, what we might call a hybrid-cloud deployment.

[4]Except for domains hosting personal sites, e.g., wordpress.com, where subdomains are ranked individually.

Rank	Domain	Total # Subdom	# EC2 Subdom
9	amazon.com	68	2
13	linkedin.com	142	3
29	163.com	181	4
35	pinterest.com	24	18
36	fc2.com	89	14
38	conduit.com	40	1
42	ask.com	97	1
47	apple.com	73	1
48	imdb.com	26	2
51	hao123.com	45	1

Table 4: Top 10 (by Alexa rank) EC2-using domains, their total number of subdomains, and the number of EC2-using subdomains.

High traffic volume domains. We complement the above with an analysis of the top domains seen in the packet capture, as measured by traffic volume. We use Bro to extract hostnames within HTTP requests and common names within the server certificates embedded in HTTPS flows[5]. Aggregating the hostnames and common names by domain, we find 13,604 unique cloud-using domains: 12,720 use EC2 and 885 use Azure. Of these 13,604 domains, 6902 were also identified as cloud-using via the Alexa dataset; the remainder were not in the Alexa top 1 million. Table 5 lists the highest 15 such domains in terms of traffic volume. A few tenants are responsible for a large fraction of the traffic. Most notably, dropbox.com accounts for almost 70% of the combined HTTP(S) traffic volume. This also explains why HTTPS (used by dropbox.com) dominates HTTP in terms of volume (though not number of flows, refer to Table 2).

It is informative to compare our analysis of top EC2-using domains by traffic volume to the analysis by DeepField networks [28], which was conducted three months before we collected the packet capture. They used data from customers of their network analysis products. Seven of the domains on our top 15 list also appear within the top 15 on DeepField's list (indicated with a (d) in Table 5).

Summary and implications. A substantial fraction of the world's most popular websites rely in whole or in part on public IaaS clouds, especially EC2. Most cloud-using domains have some subdomains hosted on a cloud service while other subdomains are hosted elsewhere. Perhaps surprisingly a small, but noticeable fraction of subdomains use both a cloud and other hosting solutions. Finally, traffic volume appears to be dominated by a few cloud tenants (we discuss traffic patterns more next). Depending on how tenants' deploy their services (e.g., how many and which regions they use), these observations have implications for availability of web-based cloud-resident services. We explore the underlying deployments in more detail in §4.

3.3 Traffic Patterns

Our packet capture enables us to analyze not only who is running on the cloud, but also the traffic patterns between clients in our university and cloud-resident web services.

Flow-level properties. We first study the number of flows observed for various cloud-using domains in the packet capture. Figures 3a and 3b show CDFs of the number of HTTP and HTTPS flows, respectively, per-domain across our entire packet capture. We observe that ≈50% of domains have fewer than 1,000 HTTP flows, and more than 80% of domains have fewer than 1,000 HTTPS flows. Upon further analysis, we found that the top 100 cloud-using domains are responsible for about 80% of the HTTP flows

[5]TLS encryption hides the hostname associated with the underlying HTTP requests, so we use the common names found in TLS server certificates as a proxy.

EC2			Azure		
Domain	**Rank**	**Traffic (%)**	**Domain**	**Rank**	**Traffic (%)**
dropbox.com (d)	119	595.0 (68.21)	atdmt.com	11,128	27.0 (3.10)
netflix.com (d)	92	14.8 (1.70)	msn.com	18	20.9 (2.39)
truste.com (d)	15,458	9.2 (1.06)	microsoft.com	31	19.7 (2.26)
channel3000.com	29,394	6.4 (0.74)	msecnd.net	4,747	13.5 (1.55)
pinterest.com (d)	35	5.1 (0.59)	s-msn.com	25,363	12.5 (1.43)
adsafeprotected.com (d)	371,837	4.7 (0.53)	live.com	7	11.8 (1.35)
zynga.com	799	3.9 (0.44)	virtualearth.net	147,025	9.2 (1.06)
sharefile.com	20,533	3.6 (0.42)	dreamspark.com	35,223	7.1 (0.81)
zoolz.com	272,006	3.2 (0.36)	hotmail.com	2,346	6.3 (0.72)
echoenabled.com (d)	-	2.7 (0.31)	mesh.com	-	4.5 (0.52)
vimeo.com	137	2.3 (0.26)	wonderwall.com	-	3.2 (0.36)
foursquare.com	615	2.2 (0.25)	msads.net	-	2.5 (0.29)
sourcefire.com	359,387	1.9 (0.22)	aspnetcdn.com	111,859	2.3 (0.26)
instagram.com (d)	75	1.5 (0.17)	windowsphone.com	1,597	2.0 (0.23)
copperegg.com	122,779	1.5 (0.17)	windowsphone-int.com	-	2.0 (0.23)

Table 5: Domains with highest HTTP(S) traffic volumes (in GB) in the packet capture. Percentages are relative to the total HTTP(S) traffic across both clouds in the capture. Domains marked with (d) appeared on DeepField's Top 15 [28].

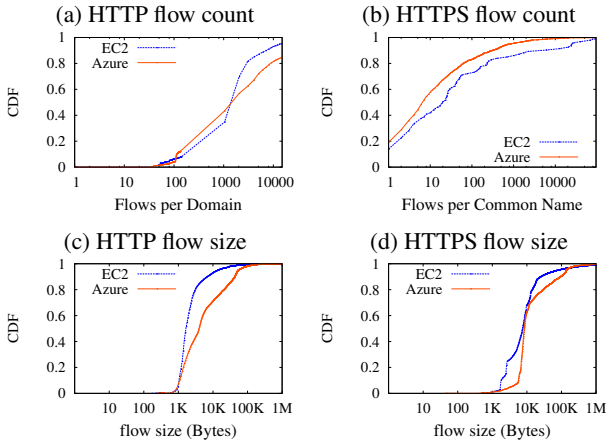


Figure 3: CDFs for HTTP/HTTPS flow counts and sizes.

Content	Bytes (GB)	%	mean (KB)	max (MB)
text/html	57.9	24.10	16	3.7
text/plain	56.2	23.37	5	24.4
image/jpeg	25.6	10.64	20	18.7
application/x-shockwave-flash	20.8	8.66	36	22.9
application/octet-stream	18.9	7.85	29	2,147
application/pdf	7.6	3.15	656	25.7
text/xml	7.5	3.10	5	4.9
image/png	7.1	2.94	6	24.9
application/zip	6.8	2.81	1,664	5,010
video/mp4	5.3	2.21	6,578	143

Table 6: HTTP content types by byte counts (in GB), as well as mean (in KB) and max (in MB) object sizes.

in EC2, and nearly 100% of the HTTP flows in Azure; a long tail follows for both clouds (CDF excluded for brevity).

Flow sizes and durations generally appear to fit heavy-tailed distributions (we omit durations in Figure 3 for brevity); similar properties have been observed for flows in other networking contexts (e.g., in data centers [21]). We note interesting differences between HTTP and HTTPS: in particular, HTTPS flows are larger and last longer than HTTP flows across both EC2 and Azure (e.g., median sizes for EC2 are 10K and 2K, respectively). This is expected given our observation that a large percentage of HTTPS traffic is from file storage services. In both cases, we see large flows that are more than a few MB in size and long flows that last for a few hours.

Content types. We next look at the types of content served by web-facing cloud tenants. We use Bro to extract the Content-Type and Content-Length fields from replies in HTTP flows. Unfortunately, we cannot extract these details for HTTPS flows because the HTTP headers are encrypted. The top content types, by byte count, for HTTP flows are shown in Table 6, along with the mean and max content sizes. About half of all content is html or plain text, and these type of objects are generally smaller in size. The majority of the remaining content is a mixture of images, Flash, generic binary data (i.e., octet-stream), PDFs, and XML; these objects are generally much larger. This suggests that EC2 and Azure HTTP traffic is primarily conveying web sites, and not (say) for file transfer. We see that some objects can be very large in size; e.g., we see binaries that are 5GB in size, and even some plain text files that are as large as 24MB.

Summary and Implications. We find that most flows arise from the top few domains. A majority of the flows are short, and HTTPS flows are generally larger in size and last longer than their HTTP counterparts. Most web services appear to be using the cloud to serve html or plain text content. These observations have implications for content delivery systems targeted toward cloud-resident services. For instance, the predominance of plain text and HTML traffic (as opposed to compressed images, binaries, videos, etc.) points to the fact that compression could be employed to save WAN bandwidth and improve content delivery latency [18].

4. TENANTS' DEPLOYMENT POSTURE

In this section, we attempt to understand *how* tenants use the cloud for deploying their front ends. We start by analyzing the deployment patterns employed by cloud-using (sub)domains. We first focus on the four patterns identified in Figure 1. We then quantify how many, and which, regions and availability zones are leveraged by cloud-using (sub)domains' front ends.

4.1 Deployment Patterns

In this section, we use the DNS records from our Alexa subdomains dataset and a variety of heuristics to detect and quantify usage of the deployment patterns outlined in Figure 1. Specifically, we estimate the use of virtual machines (VMs), platform-as-a-service (PaaS) environments, load balancers, content-distribution networks (CDNs), and domain name servers within the front ends of web services hosted in both EC2 and Azure. In general, we discuss EC2 and Azure separately because of differences in the cloud architectures.

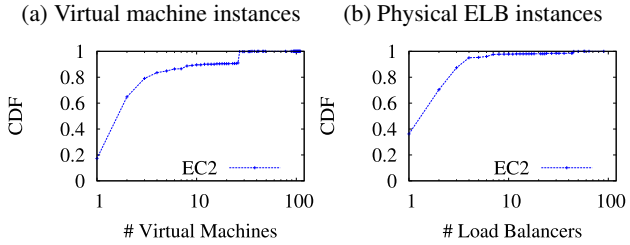


Figure 4: CDFs for the # of feature instances per subdomain (only includes subdomains which use the feature).

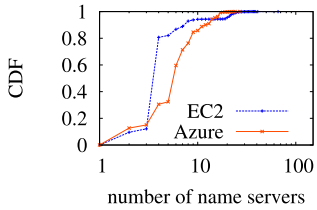


Figure 5: CDF of the # of DNS servers used per subdomain.

VM front end in EC2. Each VM instance in an IaaS cloud combines a set of virtual resources (CPU core(s), memory, local storage, and network bandwidth) whose capacity depends on the instance type. In EC2 each instance is assigned an internal IP address within a region-specific private network; EC2 tenants may optionally assign a public (i.e., Internet-routable) IP address to a VM.

We identify usage of VMs as directly-reachable web service front ends—i.e., deployment pattern *P1* (Figure 1a)—by examining if the DNS query for an EC2-using subdomain directly returns an IP address (instead of a CNAME), which we then associate with a VM instance. We find that 505,578 (72%) EC2-using subdomains leverage front end VMs. Figure 4a shows a CDF of the number of front end VM instances used by each EC2-using subdomain; this CDF only includes subdomains which use front end VMs. We observe that about half of such subdomains use 2 front end VMs and 15% use 3 or more front end VMs.

Aggregating by domain, we find that 52% of EC2-using domains have at least one subdomain which uses at least one front end VM. If we sum the number of front end VMs used across all subdomains of a given domain, we find that 10% of domains which use front end VMs in EC2 use 3 or more front end VMs in total.

PaaS front end in EC2. PaaS systems offer a hosted environment for deploying web applications, avoiding the need for tenants to manage low-level system details. PaaS systems are frequently built atop existing IaaS infrastructure: e.g., Amazon's Elastic Beanstalk [5] and Heroku [14] both run atop EC2. A Beanstalk environment always includes an Amazon Elastic Load Balancer (ELB) instance (discussed in more detail below), reflecting deployment pattern *P2* (Figure 1b, replace VMs with PaaS nodes). A Heroku environment may or may not include an ELB, reflecting usage of deployment patterns *P2* or *P3* (Figure 1), respectively.

We say that a subdomain uses Beanstalk or Heroku if the subdomain's DNS record has a CNAME that (*i*) includes 'elasticbeanstalk' or any of 'heroku.com', 'herokuapp', 'herokucom', and 'herokussl' and (*ii*) resolves to an IP in EC2's public IP address range. In the case of Heroku without ELB, the IPs to which the CNAME resolves represent PaaS nodes; we associate these IPs with the subdomain whose DNS record contains the corresponding CNAME.

A total of 201,666 (28%) EC2-using subdomains in our Alexa subdomains dataset contain a CNAME in their DNS record. Applying the above filters for PaaS, we find that 60,273 (8%) EC2-using subdomains use a front end PaaS environment in EC2. Of these, over 97% (59,991) are using Heroku; only 3% use Elastic Beanstalk. Amazon always includes an ELB in a Beanstalk environment, but Heroku only sometimes leverages ELBs—only 3% of subdomains (1,850) which use Heroku also use ELB. We therefore conclude that, in the case of EC2, PaaS systems are predominantly used according to deployment pattern *P3* (Figure 1c).

We now focus on the 58,141 (59,991 - 1,850) subdomains that use Heroku without ELB. We find that these are associated with just 94 unique IPs. Although we have no insight into the number of worker instances used by Heroku, this shows that Heroku is multiplexing PaaS functionality among a relatively large number of subdomains: in particular, we find that about one-third of subdomains using Heroku share the CNAME 'proxy.heroku.com'.

Load balancer front end in EC2. Load balancers divide traffic among a set of "worker" VMs or PaaS nodes, as reflected in deployment pattern *P2* (Figure 1b). Amazon Elastic Load Balancers (ELBs) [13] are Amazon-managed HTTP proxies. An EC2 tenant requests an ELB in a specific region and subsequently associates VM instances, in one or more zones, with this ELB. The ELB automatically round-robins requests among zones and among the VMs in each zone. In fact, traffic is routed to zone-specific ELB proxies by rotating the order of ELB proxy IPs in DNS replies. ELB can also be used with PaaS, as discussed above.

When a subdomain uses an ELB, the subdomain's DNS record contains a CNAME ending in 'elb.amazonaws.com'; the CNAMEs resolve to IP addresses for one or more ELB proxies. We identify ELB-using subdomains in our Alexa subdomains dataset based on the presence of such CNAMEs; we refer to each distinct CNAME as a "logical ELB instance". We also associate with the subdomain the IPs of the specific ELB proxies to which the CNAME resolves; we refer to these as "physical ELB instances".

We find that 27,154 (4%) EC2-using subdomains use ELB as their front end. Of the subdomains that use ELB, 280 (1%) use it in the context of Elastic Beanstalk and 1,850 (6.8%) use it with Heroku. Aggregating by domain, we find that 9,851 (26%) EC2-using domains use front end ELB(s).

Across all ELB-using subdomains, we observe 15,703 physical ELB instances (i.e., distinct IPs associated with ELB CNAMEs). Hence, while each subdomain has its own logical ELB(s), the physical ELB proxies that perform the actual load balancing appear to be shared across multiple, even unrelated, subdomains. In particular, we analyzed the number of subdomains per physical ELB and found that ≈4% of the physical ELB instances are shared by 10 or more subdomains.

Figure 4b shows a CDF of the number of physical ELB instances associated with each subdomain; this CDF only includes subdomains which use ELB. We observe that about 95% of ELB-using subdomains are associated with 5 or fewer physical ELB instances. A few ELB-using subdomains (e.g., dl.outbrain.com and m.netflix.com) use many physical ELB instances: 58 and 90, respectively.

Front ends in Azure. Azure's architecture differs from EC2 insofar as clients cannot distinguish whether a web service uses a VM, PaaS, or load balancer front end. In Azure, VMs and PaaS environments are both encompassed within logical "Cloud Services" (CS). An individual CS may contain (*i*) a single VM, (*ii*) a collection of related VMs, or (*iii*) a PaaS environment. Each CS is assigned a unique DNS name ending with 'cloudapp.net' and a corresponding public IP address. Traffic sent to this public IP goes through a transparent proxy—which performs NAT and, optionally, load balancing—before directing traffic to a VM or PaaS node. Thus, a CS may reflect deployment patterns *P1*, *P2* (with VMs or PaaS

nodes), or *P3* (Figure 1), all of which appear the same from a client perspective.

We examine the DNS records for Azure-using subdomains in the Alexa subdomains dataset to identify subdomains which use a CS (i.e., VM, PaaS, or load balancer) front end. If the DNS query for an Azure-using subdomain either directly returns an IP address or returns a CNAME ending in 'cloudapp.net', then we say the subdomain uses a CS front end. We associate the directly returned IP or the CNAME, and its corresponding IP, with a CS instance.

A total of 1,153 (17%) Azure-using subdomains directly resolve to an IP address and 5,404 (82%) Azure-using subdomains contain a CNAME in their DNS record. Applying the above filters for CS, we find that 4,581 (70%) Azure-using subdomains use a CS front end. Aggregating by domain, we find that 57% of Azure-using domains have at least one subdomain which uses a CS front end.

Azure also offers a unique feature (which has no parallel in EC2) for load balancing across front ends: Azure Traffic Manager (TM) [9] uses DNS to direct traffic to different CSs, which may be spread across multiple regions. TM can, based on a tenant's preference, do performance-based load balancing (finding the CS closest to the client), failover load balancing (picking the next active CS), or simple round-robin load balancing. When a subdomain uses TM, its DNS record contains a CNAME ending in 'trafficmanager.net', similar to ELB. However, TM performs all load balancing using DNS—unlike ELB which uses a combination of DNS and physical proxies—so TM CNAMEs resolve directly to a CNAME for a specific CS (e.g., 'abc.cloudapp.net'). We find that only 100 (2%) Azure-using subdomains (corresponding to 52 domains) use TM.

The aforementioned CNAME-based filters for ELB, Beanstalk, Heroku, CS, and TM were not applicable to 116,323 (16%) EC2-using subdomains, and 1,938 (30%) Azure-using subdomains. We are investigating techniques to understand the deployment patterns underlying these subdomains.

Content distribution networks. We now focus on the use of CDNs, which we illustrated in deployment pattern *P4* (Figure 1d). Note that CDNs can be employed alongside any of the other three deployment patterns.

Both Microsoft and Amazon run their own CDNs, which we focus on studying. Amazon's CloudFront CDN [3] uses a different public IP address range than the rest of EC2. Hence, we determine if a subdomain uses CloudFront by observing if its DNS records contain one or more IPs in CloudFront's IP range. Azure's CDN [7] uses the same IP address ranges as other parts of Azure, so we detect whether a subdomain uses the Azure CDN based on whether a subdomain's DNS records contain CNAMEs with 'msecnd.net'.

We find 7,622 subdomains (corresponding to 5,988 domains) use CloudFront and 68 subdomains (corresponding to 54 domains) use Azure's CDN. Despite the much smaller number of domains using Azure's CDN, there is still a significant volume of traffic associated with msecnd.net in our packet capture dataset (Table 5). Azure's CDN is clearly being used within some Microsoft properties, perhaps to host embedded content or cookies.

Domain name servers. The first step in accessing a cloud-resident service is to resolve its name (Figure 1). In what follows, we examine cloud-resident subdomain's use of DNS, focusing on the extent to which they rely on cloud providers for DNS services as well.

We identifed the "location" of a cloud-using subdomain's authoritative name server(s) as follows: For each DNS record associated with a given subdomain in our Alexa subdomains dataset, we extract all the domains specified in the NS records. We then performed a DNS lookup on each of these domains from 50 globally-distributed PlanetLab nodes. We flushed and reset the cache of the local resolver between each DNS lookup, and we added the 'norecurse' flag to each DNS query to minimize the influence of caching. We compare the resulting IP addresses to the public IP address ranges for EC2, CloudFront, and Azure.

Cloud	Feature	# Domains	# Subdomains	# Inst.
EC2	VM	19.9K (52.5%)	505.6K (71.5%)	28.3K
	ELB	9.9K (25.9%)	27.1K (3.8%)	15.7K
	BeanStalk (w/ ELB)	188 (0.5%)	280 (< 0.01%)	455
	Heroku (w/ ELB	622 (1.6%)	1.9K (0.3%)	2.4K
	Heroku (no ELB)	1.3K (3.5%)	58.1K (8.2%)	94
Azure	CS	863 (37.0%)	4.5K (68.3%)	790
	TM	52 (2.2%)	100 (1.5%)	78

Table 7: Summary of cloud feature usage.

Rank	Domain	# Cloud Subdom	Front-end VM	Front-end PaaS	Front-end ELB	ELB IPs	Use CDN
9	amazon.com	2	0	1	2	27	0
13	linkedin.com	3	0	1	1	1	0
29	163.com	4	0	0	0	0	4*
35	pinterest.com	18	4	0	0	0	0
36	fc2.com	14	10	0	4	68	0
38	conduit.com	1	0	1	1	3	0
42	ask.com	1	1	0	0	0	0
47	apple.com	1	1	0	0	0	0
48	imdb.com	2	0	0	0	0	1
51	hao123.com	1	0	0	0	0	1*

Table 8: Cloud feature usage for the highest ranked EC2-using domains (* indicates use of a CDN other than CloudFront).

We observe a total of 23,111 name servers supporting the 713K cloud-using subdomains in our Alexa subdomains dataset. Many subdomains use the same name servers, leading to a smaller set of name servers than subdomains. Figure 5 shows a CDF for the number of name servers used by each cloud-using subdomain; we observe that nearly 80% of subdomains use 3 to 10 name servers. We categorize the name servers as follows: 2,062 were hosted in CloudFront, which appears to host Amazon's route53 DNS service as many of these name servers had 'route53' in their domain name; 1,239 were running inside EC2 VM instances; 22 were hosted inside Azure VM instances or Azure CS; and 19,788 were hosted outside any of EC2, CloudFront, or Azure;

The above analyses are summarized in Table 7 which shows how many (sub)domains in our Alexa subdomains dataset use each cloud feature. We also show the number of instances (identified by IP address) of that feature.

Analysis of top domains. As notable exemplars, Table 8 gives a detailed breakdown of the cloud feature usage of the most popular (according to Alexa rankings) EC2-using domains. We observe that the majority of subdomains associated with the top domains have VM or ELB front ends. Of those using ELB front ends, amazon.com and fc2.com use ELB the most (i.e., there are more physical ELB IPs associated with these domains). Three of the top domains have subdomains which use a CDN, but only one of these domains uses the CloudFront CDN.

Summary and implications. In summary, we find that the majority (71.5%) of EC2-using subdomains use a VM front end (deployment pattern *P1*); hence most EC2 tenants are using EC2 as a true IaaS cloud. Only a small fraction use an ELB front end (3.8%) or PaaS front end (8.5%). Due to limited use, failures of value-added features are unlikely to have a major impact on EC2-using subdomains. In Azure, we are able to identify the usage of VM, PaaS, or load balancing front ends (we cannot distinguish which) for 70% of subdomains. A small fraction (1.5%) of Azure-using domains

Region	Location	# Dom	# Subdom
ec2.us-east-1	Virginia, USA	25,722	521,681
ec2.eu-west-1	Ireland	6,834	116,366
ec2.us-west-1	N. California, USA	3,950	40,548
ec2.us-west-2	Oregon, USA	1,548	15,635
ec2.ap-southeast-1	Singapore	1,800	20,871
ec2.ap-northeast-1	Tokyo, Japan	2,223	16,965
ec2.sa-east-1	São Paulo, Brazil	625	14,866
ec2.ap-southeast-2	Sydney, Australia	313	554
az.us-east	Virginia, USA	268	862
az.us-west	California, USA	161	558
az.us-north	Illinois, USA	590	2,071
az.us-south	Texas, USA	1,072	1,395
az.eu-west	Ireland	564	1,035
az.eu-north	Netherlands	573	1,205
az.ap-southeast	Singapore	379	632
az.ap-east	Hong Kong	333	502

Table 9: EC2 and Azure region usage Alexa subdomains

leverage TM to balance traffic across different front ends. The majority of DNS servers used by cloud-using subdomains reside outside of EC2 or Azure, giving subdomains the option of routing traffic to different resources (in another cloud or a private data center) in the event of cloud failure.

4.2 Region Usage

EC2 and Azure give tenants the choice of using one or more geographically distinct regions (i.e., data centers). Regions provide a mechanism for robustness in the case of catastrophic failures, e.g., regional power or service outages [4, 6, 24, 35]. In this section, we examine how many, and which, of the eight regions offered by each cloud provider are leveraged by the front ends of cloud-using (sub)domains.

We ascertain the region(s) used by each subdomain in the Alexa subdomains dataset by comparing the IP addresses associated with that subdomain against the per-region IP address ranges published by EC2 [12] and Azure [8]. We only consider IPs associated with VM, PaaS, and ELB/TM.

Figure 6a shows a CDF (note the Y-axis starts at 90%) of the number of regions used by each subdomain in the Alexa subdomains. Over 97% of EC2-using and 92% of Azure-using subdomains exclusively use one region. Across all domains (Figure 6b), the trend of low region usage is largely the same, although, the fraction of Azure-using domains that only use one region (83%) is smaller than the fraction of subdomains that only use one region (92%).

The number of (sub)domains (from the Alexa subdomains) in each region are shown in Table 9. We observe that the usage of EC2 regions is heavily skewed towards a few regions: 74% of EC2-using subdomains use *US East* and 16% use *Europe West*. Azure, relatively speaking, has a more even distribution of subdomains across regions, but each region has significantly fewer subdomains. The most used Azure regions are *US South* and *US North*.

Analysis of top domains. We now focus on region usage of subdomains corresponding to the most popular (according to Alexa rankings) domains. Our analysis is summarized in Table 10. As with the rest of our results above, we see that in all but two cases, subdomains appear to use a single region. The exceptions are msn.com and microsoft.com, where 11 of the 89 subdomains and 4 of 11 subdomains, respectively, use two regions each. No popular subdomain uses three or more regions. We also note that in some cases, a domain may deploy different subdomains across different regions: e.g., live.com's 18 subdomains are spread across 3 regions. Contrarily, there are domains whose subdomains are all deployed in one region (e.g., pinterest.com).

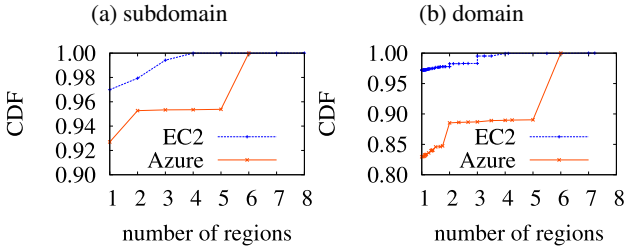


Figure 6: (a) CDF of the number of regions used by each subdomain (b) CDF of the average number of regions used by the subdomains of each domain.

Rank	Domain	# Cloud Subdom	Total # Regions	k=1	k=2
7	live.com	18	3	18	0
9	amazon.com	2	1	2	0
13	linkedin.com	3	2	3	0
18	msn.com	89	5	78	11
20	bing.com	1	1	1	0
29	163.com	4	1	4	0
31	microsoft.com	11	5	7	4
35	pinterest.com	18	1	18	0
36	fc2.com	14	2	14	0
42	ask.com	1	1	1	0
47	apple.com	1	1	1	0
48	imdb.com	2	1	2	0
51	hao123.com	1	1	1	0
59	go.com	4	1	4	0

Table 10: Region usage for the top cloud-using domains. The third column is the number of cloud-using subdomains; fourth is the total number of regions used by a domain; and the $k = 1$ and $k = 2$ columns are the number of subdomains which use one or two regions, respectively.

Analysis of subdomain deployment vs. customer location. An interesting question about cloud service deployment is whether subdomains are deployed near their customers? The answer to this question reveals whether current cloud services are deployed in an "optimal" manner, because deploying a service near customers usually leads to better client network performance (lower latency and higher throughput).

To answer this question, we leverage the client geo-location information provided by the Alexa web information service [2]. For example, at the time of writing, Alexa reported that 47% of clients accessing pinterest.com are from the United States, 10.4% from India, 3.2% from the United Kingdom, 3.1% from Canada, and 2.1% from Brazil. For each domain, we define the "customer country" as the country where the largest fraction of clients are located. We assume the customer country is the same for all of a website's subdomains, as Alexa does not track subdomains separately. For instance, the United States is the customer country for pinterest.com (and its subdomains) based on our definition.

We performed the analysis for all of the cloud-using subdomains (about 713K) in our dataset. Our measurement methodology was able to successfully identify approximately 538K (75% of the total) subdomains' customer country. We find that 252K (47%) subdomains' customer country is not the same as the country where this subdomain is hosted. Moreover, 174K (32%) subdomains are not even hosted on the same continent as the subdomains' customer country. This implies that a large fraction of web services are prob-

Instance type	Zone of probe destination		
	ec2.us-east-1a	ec2.us-east-1c	ec2.us-east-1d
t1.micro	0.6 / 0.6	1.4 / 1.5	1.9 / 2.0
m1.medium	0.5 / 0.6	1.5 / 1.6	2.0 / 2.1
m1.xlarge	0.4 / 0.5	1.4 / 1.5	1.8 / 1.9
m3.2xlarge	0.4 / 0.5	1.5 / 1.7	1.9 / 2.0

Table 11: RTTs (least / median) in milliseconds over 10 probes from a micro instance in ec2.us-east-1a to an instance with a certain type (rows) and zone (columns).

ably not deployed in an optimal manner in terms of network performance. We suspect that the current deployment posture is affected by computing, storage, and network costs and/or how long the cloud region has existed. In §5, we explore how much opportunity exists for improving wide-area performance through changes in region usage.

Summary and implications. Our key finding in this section is that most popular domains and subdomains appear to be using a single region. This has significant implications on both the robustness and performance of cloud-using web services. From an availability perspective, an outage of EC2's *US East* region would take down critical components of at least 2.3% of the domains (61% of EC2-using domains) on Alexa's list of the top 1 million websites. This is a lower bound, as our results do not include dependencies between domains. From a performance perspective, our analysis of web service deployment and customer locations reveals that a considerable fraction of client traffic may travel farther than necessary due to suboptimal provisioning.

4.3 Availability Zone Usage

Within each region of EC2, cloud tenants have the choice of deploying across multiple zones. EC2 zones offer a means for improving service robustness as they are claimed to use separate compute, network, and power infrastructure so that a failure in any of these will not affect more than one zone. There seems to be no equivalent of zones in Azure.

We now focus on determining the zone deployment for EC2-using services' front ends. Unlike the regions, which are easily distinguished based on the IP address of a subdomain and the advertised ranges [8, 12], there is no direct way to associate an IP address to a zone. We therefore turn to cloud cartography techniques [34]. We use two methods to identify zones: network latency and proximity in the internal addresses to instances with a known zone (i.e., VMs we launched).

Latency-based identification. The first technique (originally used in [34]) aims to estimate whether a target physical instance (e.g., VM instance, physical ELB instance, etc.) is in the same zone as an instance under our control by measuring the RTT to the target. The RTT will be significantly smaller when both instances are in the same zone, compared to when they are in different zones, presumably reflecting a longer path in the latter case. We performed a simple experiment to confirm that RTTs follow this trend. We setup an m1.micro instance in the ec2.us-east-1a zone and measured the RTT (over 10 trials) to one instance of each type in each of the three zones ec2.us-east-1a, ec2.us-east-1c, and ec2.us-east-1d. Table 11 shows both the minimum and median RTTs. It is clear that the instances in the same availability zone have the smallest RTTs (about 0.5ms) regardless of the instance type. We repeated this experiment in each of the EC2 regions with similar results.

Our experiment corroborates similar previous ones [34] regarding the efficacy of latency-based zone estimates. However, there are several complicating factors when attempting to deploy this methodology for zone measurement in practice. First, our and prior experiments used a limited number of otherwise idle EC2 instances—RTT times may be significantly more noisy across a broader set of instances and for more heavily loaded instances. Second, in some regions (ec2.us-east-1) we are unable to run an instance in each zone. Nevertheless, we perform, to the best of our knowledge, the first use of latency-based zone measurement at scale.

Region	# tgt IPs	# resp.	1st zn	2nd zn	3rd zn	% unk
ec2.us-east-1	34,194	25,085	11,592	2,835	10,658	16.6
ec2.us-west-1	3,663	2,471	1,050	1,367	N/A	32.5
ec2.us-west-2	1,869	1,679	600	755	324	10.1
ec2.eu-west-1	8,581	7,023	1,935	2,095	2,993	18.2
ec2.ap-northeast-1	2,558	1,260	1,129	N/A	131	50.7
ec2.ap-southeast-1	2,296	1,987	968	1,019	N/A	13.5
ec2.ap-southeast-2	333	298	146	152	N/A	10.5
ec2.sa-east-1	701	616	376	240	N/A	12.1

Table 12: Estimated distribution of instance IPs across zones using latency method ($T = 1.1$). Second column is total number of IPs derived from subdomains, followed by the number that responded to probes, and the estimates of how many were in each of the zones. The final column is the percentage of the responding IPs for which no zone could be estimated.

For each region, we launch three m1.medium instances in each zone from which to perform probes; we refer to these as probe instances. In ec2.us-east-1, we use ten additional m1.small instances in each zone because a large fraction of IPs are in this region. This region proved more challenging, due to a higher density of instances, a lack of full coverage of zones by probe instances, and more network noise. For each target physical instance in a region, we first map the public IP address to an internal IP address via an appropriate DNS query from a probe instance in that region. We then use hping3 to perform 10 TCP pings from a probe instance in each zone in the region to both the internal IP address and the public IP. While probes are very lightweight, we nevertheless limited the frequency of our probing, and the probing process was repeated 5 times on each probe instance. The experiment was performed over the course of five days (April 4th to April 8th, 2013). The minimal RTT is taken as the probe time. For a region with k zones (for ec2.us-east-1 we have $k = 3$, even though it has five zones), we end up with k probe times $t_1, \ldots, t_k$ and we let i be such that $t_i < t_j$ for all $i \neq j \in [1 \,..\, k]$. If there exists no such i (due to a tie), then we mark the target IP (physical instance) as having unknown zone. If t_i is less than a threshold T then we conclude that the target physical instance is in zone i. Otherwise, we mark the target as having an unknown zone.

Setting $T = 1.1$, we end up with a zone estimate for physical instance IPs from our Alexa subdomains dataset in most regions. The results are shown in Table 12. The technique worked well for all regions except for ec2.ap-northeast-1. The unknown rate is affected by two factors: (*i*) Whether we can set up instances in all zones; for example, we can not set up instances in ec2.ap-northeast-1's zone #2 after January, 2013, but, according to our observation in January, the number of IPs in zone #1 and zone #2 is quite similar. (*ii*) How many times we repeat the probes to reduce network noise; with more probe data, the unknown rate can be further reduced.

Address-proximity-based identification. We supplement the latency measurements with sampling using our own accounts and an estimation mechanism based on proximity of a target internal IP address to a sampled IP address. As shown in prior work [34], it is very likely that two instances running in the same /16 subnet are

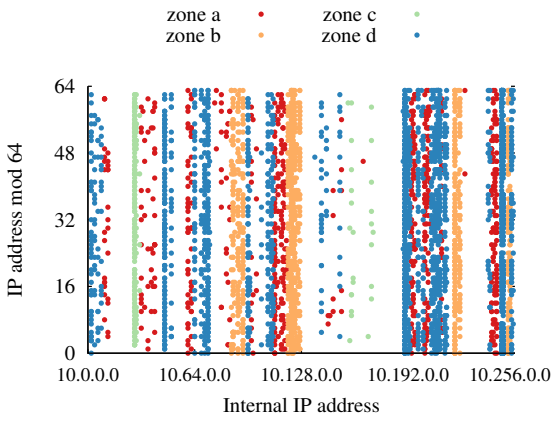


Figure 7: Sampling data for address proximity measurement.

co-located in the same zone (and are potentially even of the same instance type). We launched 5096 instances (in aggregate over the course of several years) under a number of our AWS accounts. The result is a set of account, zone label, internal IP triples (a_i, z_i, ip_i) for $i \in [1 .. X]$. A complicating factor is that, for $a_i \neq a_j$ (different accounts), it may be that the EC2-provided zone labels are not the same. Meaning, for account a_i it may be that the ec2.us-east-1a is not the same actual zone as ec2.us-east-1a for account a_j. Let Z be the set of zone labels.

We thus take the following straightforward approach to merge data across multiple different accounts. Consider a pair of accounts a, b. Find the permutation $\pi_{b\to a}$: $Z \to Z$ that maximizes the number of pairs of /16 IPs $ip_i/16 = ip_j/16$ such that $a_i = a$, $a_j = b$ and $\pi_{b\to a}(z_j) = z_i$. This can be done efficiently by ordering all triples of the accounts a and b by IP address, and inspecting the zone labels associated to each account for nearby IP addresses. One can repeat this for all pairs of accounts and solve the integer programming problem associated with finding an optimal set of permutations π, but it proved effective to take the simpler approach of finding $\pi_{b\to a}$ for one pair, merging the pair's triples by applying $\pi_{b\to a}$ appropriately, and then repeating with the next account c, etc. The outcome of applying this process to samples from ec2.us-east-1 is shown in Figure 7. Each binned IP address is a point, with the distinct colors representing distinct availability zones.

We now apply the sampling data to the physical instances from the Alexa subdomains dataset. If we have at least one sample IP in the same /16 subnet as the IP associated with a target physical instance, we conclude that the target instance is in the same zone as the sample instance. Otherwise, we mark the target instance as having an unknown zone. With this approach, we are able to identify the zone for 79.1% of the EC2 physical instances in the Alexa subdomains dataset.

Treating these zone identifications as ground truth, we check the accuracy of the latency-based zone identifications. Table 13 shows for each EC2 region the total number of physical instances in the Alexa subdomains dataset, the number of instances for which the two zone identification approaches agree, the number of instances whose zone cannot be identified using one or both methods, the number of instances where the two methods disagree, and the error rate of the latency-based method. The error rate is defined as the number of mismatched instances / (the total number of instances - the number of unknown instances). We observe that latency based method's overall error rate is 5.7%. Its error rate is less than 3.9%

Region	count	match	unknown	mismat.	error rate
all	37876	28640	7494	1742	5.7%
ec2.ap-northeast-1	1260	965	295	0	0.0%
ec2.ap-southeast-1	1987	1558	428	1	<0.1%
ec2.ap-southeast-2	298	201	97	0	0.0%
ec2.eu-west-1	6102	3359	1597	1146	25.0%
ec2.sa-east-1	616	0	616	0	N/A
ec2.us-east-1	23518	19228	3748	542	2.7%
ec2.us-west-1	2417	2032	385	0	0.0%
ec2.us-west-2	1678	1297	328	53	3.9%

Table 13: Veracity of latency-based zone identification.

Region	1st zone		2nd zone		3rd zone	
	#Dom	#Sub	#Dom	#Sub	#Dom	#Sub
ec2.us-east-1	16.1	419.0	6.2	155.4	9.5	292.9
ec2.us-west-1	1.6	33.2	3.0	37.4	N/A	N/A
ec2.us-west-2	0.9	13.4	1.0	9.6	0.8	7.3
ec2.eu-west-1	2.3	77.0	2.9	63.9	4.5	98.7
ec2.ap-northeast-1	0.4	3.7	1.3	11.3	1.5	12.9
ec2.ap-southeast-1	0.9	11.3	1.2	19.1	N/A	N/A
ec2.ap-southeast-2	0.2	0.3	0.2	0.3	N/A	N/A
ec2.sa-east-1	0.5	14.4	0.2	8.9	N/A	N/A

Table 14: Estimated number of domains and subdomains using various EC2 zones. Some regions only have 2 zones.

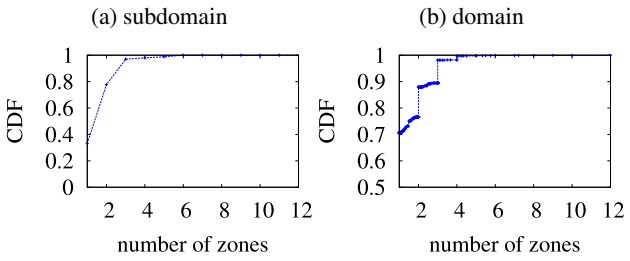


Figure 8: (a) CDF of the number of zones used by each subdomain (b) CDF of the average number of zones used by the subdomains of each domain.

for all regions except Europe West[6]. In particular, the error rate in the US East region (where the majority of the instances reside) is quite low (2.7%).

Combined identification. We combine the two zone identification methods to maximize the fraction of physical instances whose zone we can identify. We give preference to our address-proximity-based zone identifications, and use our latency-based identifications only for instances whose zone cannot be identified using the former method. Combining the two methods allows us to identify the EC2 availability zone for 87.0% of all physical EC2 instances in the Alexa subdomains dataset.

Table 14 summarizes the number of (sub)domains using each region and zone. In all but one region (Asia Pacific Southeast 2), we observe a skew in the number of subdomains using each zone in a region. Asia Pacific Northeast and US East regions have the highest skew across their three zones: 71% and 63% fewer subdomains, respectively, use the least popular zone in those regions compared to the most popular zone.

We also look at the number of zones used by each (sub)do-main. Figure 8a shows a CDF of the number of zones used by each subdomain. We observe that 33.2% of subdomains use only one zone, 44.5% of subdomains use two zones, and 22.3% of subdomains use

[6]We were unable to decrease the error rate for Europe West even after gathering additional latency measurements.

Rank	domain	# subdom	# zone	k=1	k=2	k=3
9	amazon.com	2	4	0	0	2
13	linkedin.com	3	5	1	1	1
29	163.com	4	1	4	0	0
35	pinterest.com	18	3	10	0	8
36	fc2.com	14	5	1	11	2
38	conduit.com	1	2	0	1	0
42	ask.com	1	1	1	0	0
47	apple.com	1	1	1	0	0
48	imdb.com	2	1	2	0	0
51	hao123.com	1	1	1	0	0

Table 15: Zone usage estimates for top using zones. Column 4 is estimated total number of zones used by all subdomains. Columns 4–6 indicate the estimated number of subdomains that use k different zones.

three or more zones. Of the subdomains that use two or more zones, only 3.1% use zones in more than one region. Figure 8b shows the average number of zones used by the subdomains of each domain. We observe that most domains (70%) only use one zone for all subdomains; only 12% of domains use two or more zones per subdomain on average.

Even for the top EC2-using domains, a large fraction of their subdomains only use a single zone (Table 15). For example, 56% of pinterest.com's EC2-using subdomains and 33% of linkedin.com's are only deployed in one zone.

Summary and implications. Our two key findings in this section are that (*i*) the majority of EC2-using subdomains only use one (33.2%) or two (44.5%) zones, and (*ii*) the subdomains using a given EC2 region are not evenly spread across the availability zones in that region. The former implies that many EC2-using subdomains would be completely unavailable if a single zone failed, and many others would be severely crippled: e.g., a failure of ec2.us-east-1a would cause 16.1% of subdomains to be completely unavailable. Our later key finding implies that an outage of a particular zone in a region may have a greater negative impact than an outage of a different zone in the same region: e.g., a failure of ec2.us-east-1a would impact ≈419K subdomains, while a failure of ec2.us-east-1b would only impact ≈155K.

5. WIDE-AREA PERFORMANCE AND FAULT TOLERANCE

Our results in the last section revealed that several services, even highly ranked Alexa domains, appear to use only a single region or even just a single availability zone. In this section, we explore the impact of these choices on web services' wide-area performance and tolerance to failures. We focus on EC2-using web services.

5.1 Wide-area Performance

The choice of region(s) by a cloud service may impact performance in at least two ways. First, clients' geo-distribution may be poorly matched to particular regions; such clients may experience poor latency and throughput compared to a more judicious deployment. Second, there could be temporary changes in which region performs best for a client due to congestion [19] or routing problems [36].

While the impact of diverse deployment of services (e.g., via CDNs) has been previously studied in other settings [27], we are unaware of any studies that assess its impact for the available diversity of modern public IaaS clouds. We therefore perform measurements to help us answer the following two questions: (*i*) To what extent does the choice of region impact performance experienced by clients of a web service? (*ii*) To what extent does the use

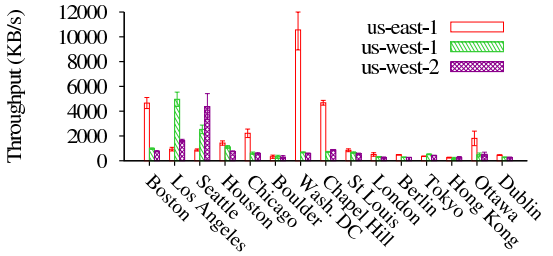


Figure 9: Average throughput between representative clients and EC2 regions in the US.

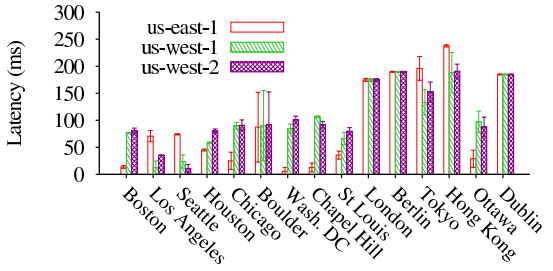


Figure 10: Average latency between representative clients and EC2 regions in the US.

of multiple regions (or zones) improve the client-perceived performance?

Latency measurements. To study per-region latency performance, we set up 40 m1.medium instances, 2 in each of the 20 availability zones available to us on EC2. We selected 80 geographically distributed PlanetLab [17] nodes as stand-ins for real clients and we used the hping3 utility to conduct 5 TCP pings to each of the 40 instances, from which we derive the average RTT. Pings that timed out were excluded from the calculations of the average. Probing was performed once every 15 minutes for three consecutive days.

Throughput measurements. We used the same set of 40 m1.medium EC2 instances and 80 PlanetLab nodes to measure throughput. We divided the PlanetLab nodes into two groups of 40. Each node in each group performed an HTTP get of a 2 MB file to one of the 40 EC2 instances (which were running Apache web server). At any given time, only one HTTP connection was established with each EC2 instance to avoid contention across throughput measurements. In particular, the clients in each group performed an HTTP get operation every 11.25 seconds; the download was canceled if it took more than 10 seconds. Each client accessed all 40 servers in each round, which means it took 450 seconds for one group to finish a round of downloading the file from each of the servers. So, in total, it took 15 minutes for 80 clients to perform one round of throughput measurements. The final throughput is measured as file_size / download_time. We ran the measurements for three consecutive days, for a total of 288 data points per client. The throughput measurements were intermingled with the latency measurements.

Performance across different regions. Figures 9 and 10 show the latency and throughput measurements for 15 representative PlanetLab locations and for the three US EC2 regions. The PlanetLab nodes are spread across the US and other parts of the world. We make a few key observations: (*i*) Single-region deployments must carefully choose a region. For example, the two US west regions do not offer equivalent "across-the-board" performance, with ec2.us-west-1 offering better average latency and throughput (130 ms and 1143 KB/s) than ec2.us-west-2 (145 ms and 895 KB/s) (averaged across all client locations). (*ii*) The charts show that the region

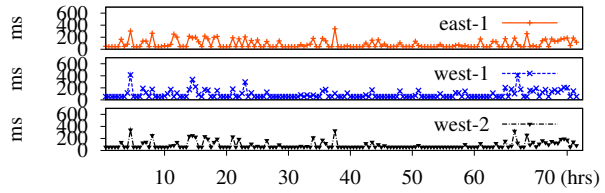


Figure 11: Latencies between Boulder site and three EC2 US regions. The best performing region changes over time.

chosen to serve content to a given client can have a significant performance impact. For example, for the client in Seattle, using the ec2.us-west-2 region can reduce latency by close to a factor of 6 and improve throughput by close to a factor of 5 compared to using ec2.us-east-1. (*iii*) We also note that the way a region is chosen may depend on the client's location: always choosing ec2.us-west-1 for the Seattle client is a good idea, but for the client in Boulder, the best choice of region may change dynamically (see Figure 11).

We now examine the relative benefits of, and choices underlying, multi-region deployments in more detail. We start by deriving an upper bound on the performance from a k-region deployment of a service. To do this, we determine the best k regions out of the 8, for $1 \leq k \leq 8$. Using our measurement data we determine the overall performance that the clients would have achieved given a routing algorithm that picked the optimal region from the k for each client and at each point in time. More specifically, for each value of k we: (*i*) enumerate all size-k subsets of regions; (*ii*) for each size-k subset compute the average performance across all clients assuming each client uses the lowest latency or highest throughput region of the k at each time round (15 min); and (*iii*) choose from these the size-k subset with lowest latency or highest throughput. Figure 12 shows the results. We find that while average performance can be increased a significant amount by adding more regions to one's deployment, there is evidence of diminishing returns after $k = 3$. In particular, latency decreases by 33% when $k = 3$ compared to $k = 1$, but only decreases by 39% when $k = 4$.

We now examine what constitutes the best k regions to use. The choice of best regions, by throughput, is as follows: ec2.us-east-1 ($k = 1$); ec2.us-east-1,ec2.eu-west-1 ($k = 2$); ec2.us-east-1, ec2.eu-west-1, ec2.us-west-1 ($k = 3$); and ec2.us-east-1, ec2.eu-west-1, ec2.us-west-1, ec2.ap-southeast-1 ($k = 4$). The choice of best regions, by latency, is: ec2.us-east-1 ($k = 1$); ec2.us-east-1,ec2.ap-northeast-1 ($k = 2$); ec2.us-east-1, ec2.ap-northeast-1, ec2.us-west-1 ($k = 3$); and ec2.us-east-1, ec2.ap-northeast-1, ec2.us-west-1, ec2.ap-southeast-1 ($k = 4$).

Performance across different zones. We also investigated the difference in performance should one use different zones in the same region. We found that the zone has little impact on latency, with almost equivalent average RTTs for all clients across the two days (results omitted for brevity). For throughput, the variation appears to be somewhat higher, but not as significant as that seen across regions. We believe such variation is due to local effects, such as contention on shared instances or network switches. This is suggested as well by other recent measurements of EC2 performance variability [25]. Moreover, in the next section we show that the Internet path variability between zones is low as well.

Summary and Implications. We find that using multiple regions can improve latency and throughput significantly. However, leveraging multiple regions may not be easy: while a given region always offers best performance for some clients, the choice of region for other clients will have to adapt in an online dynamic fashion.

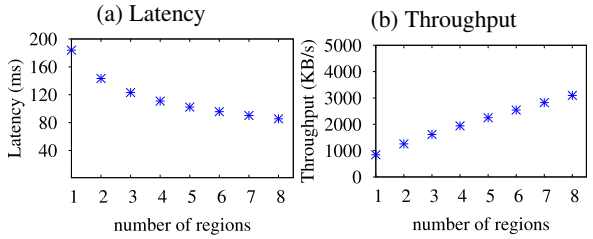


Figure 12: Average latency/throughput across all clients using an optimal k-region deployment.

Region	AZ1	AZ2	AZ3
ec2.us-east-1	36	36	34
ec2.us-west-1	18	19	n/a
ec2.us-west-2	19	19	19
ec2.eu-west-1	10	11	13
ec2.ap-northeast-1	9	n/a	9
ec2.ap-southeast-1	11	12	n/a
ec2.ap-southeast-2	4	4	n/a
ec2.sa-east-1	4	4	n/a

Table 16: Number of downstream ISPs for each EC2 region and zone.

This could be achieved via global request scheduling (effective, but complex) or requesting from multiple regions in parallel (simple, but increases server load).

While a multiple region deployment helps improve web service performance and protects against major cloud failures, cloud tenants must also consider other factors in making their decision of how many and which regions to use. First, cloud providers charge for inter-region network traffic, potentially causing tenants to incur additional charges when switching to a multi-region deployment. Second, the design of particular cloud features may restrict how a tenant's data can be shared across regions: e.g., objects stored in Amazon's Simple Storage Service (S3) can only be stored in one region at a time and Amazon Machine Images (AMIs) cannot be shared between regions. Lastly, deployments that rely on fewer features may be less susceptible to failures—e.g., deployments which only use VMs, and not other services like Amazon Elastic Block Storage or Amazon ELB, have not been affected by some major outages [4, 6]—reducing the need for resiliency through the use of multiple regions.

5.2 ISP Diversity

We now investigate tolerance to wide-area faults. Having in previous sections already established the reliance of many cloud-using services on one zone or region, we now focus on diversity in the immediate downstream ISPs at each EC2 zone. Greater diversity, and an even spread of routes across downstream ISPs, generally indicates greater tolerance to failures in Internet routing.

To do this study, we set up three m1.medium instances in each of the available EC2 availability zones. Then we ran *traceroute* 50 times from each of these instances to each of 200 geographically diverse PlanetLab nodes (Figure 2). Finally, we used the UNIX 'whois' utility to determine the autonomous system (AS) number associated with the first non-EC2 hop, and we count that AS as an immediate downstream ISP for the zone hosting the instance. The discovered ASs constitute a lower bound for the true number of ASs. Table 16 gives the number of distinct ASs seen for each zone and region. We note that: (*i*) different zones in a region have (almost) the same number of downstream ISPs; and (*ii*) the extent of diversity varies across regions, with some connected to more than 30 downstream ISPs and others connected to just 4. Except

for South America and Asia Pacific Sydney, other regions are well-multihomed.

We also studied the spread of routes across downstream ISPs (not shown). We found it to be rather uneven: even when using well-connected regions – e.g., ec2.us-west-1 and ec2.eu-west-1– we found that up to 31% (ec2.us-west-1) and 33% (ec2.eu-west-1) of routes use the same downstream ISP.

Summary and Implications. Although individual regions are multihomed, the uneven spread of routes implies that local failures in downstream ISPs can cause availability problems for large fractions of clients of cloud-using web services. This could be overcome by using multiple regions at once, or by leveraging dynamic route control solutions [20].

6. RELATED WORK

There have been several studies of the network characteristics of data centers (DCs). For example, Kandula et al. [26] focus on measuring the flow level characteristics. Benson et. al [21] studied three types of DCs: universities, private enterprise and public cloud. They characterize the internal network traffic, and infer the applications deployed in the DCs as well. Similarly, Mishra et al. [32] characterize cloud backend workloads inside Google compute clusters. Li et al. [29] benchmark and compare the computing and storage service's performance across different public cloud providers. They also measure the network bandwidth between DCs. In roughly the same vein, Chen et al. [22] study the wide-area traffic patterns between Yahoo! DCs. Our work, in contrast, is more "user-facing" than the above, in that we focus on characterizing deployment patterns of the front ends of cloud-using web services, the wide-area traffic they impose, and the implications of these properties on client perceived performance and availability. Our work therefore extends prior works on DC and cloud workload characterization along important new axes.

7. CONCLUSION

In this work we performed the first extensive measurement study of usage of modern infrastructure-as-a-service (IaaS) clouds, in particular Amazon EC2 and Windows Azure. This measurement study, which combined data from a university packet capture, interrogation of DNS records for websites listed on the Alexa top 1 million list, and lightweight probing, confirms the oft-reported anecdotes about the extent to which modern web services use these cloud systems. We profile the deployment patterns observed for popular web services. We uncover that, in many ways, these deployments are somewhat precarious, as the vast majority of (even very popular) websites use only one cloud region. In addition to resiliency benefits, we show that multi-region deployments could significantly increase latency and throughput performance in EC2. Beyond providing a snapshot on the state of cloud usage, we believe our work will spark further research on tracking cloud usage, as well as developing cloud-region-aware routing and deployment mechanisms. Finally, we make all data sets used in this paper publicly available [10], with the exception of the packet capture.

8. ACKNOWLEDGMENTS

We would like to thank Dale Carder, Michael Hare and Mark Tinberg from the Department of Information Technology at the of University of Wisconsin-Madison for helping capture packet traces. We would also like to thank our shepherd and the anonymous reviewers for their insightful feedback. This work is supported by NSF awards 1040757 and 1065134.

9. REFERENCES

[1] Alexa. http://alexa.com/topsites.

[2] Alexa Web Information Service. http://aws.amazon.com/awis/.

[3] Amazon CloudFront. http://aws.amazon.com/cloudfront.

[4] Amazon ELB Service Event in the US-East Region. http://aws.amazon.com/message/680587/.

[5] AWS Elastic Beanstalk. http://aws.amazon.com/elasticbeanstalk.

[6] AWS Service Event in the US-East Region. https://aws.amazon.com/message/680342/.

[7] Azure CDN. http://msdn.microsoft.com/en-us/library/windowsazure/ee795176.aspx.

[8] Azure Datacenter IP Ranges. http://microsoft.com/en-us/download/details.aspx?id=29840.

[9] Azure TM. http://msdn.microsoft.com/en-us/library/windowsazure/hh744833.aspx.

[10] Cloud Measurement Data. http://pages.cs.wisc.edu/~keqhe/cloudmeasure_datasets.html.

[11] Cloudflare CDN. http://cloudflare.com.

[12] EC2 public IP ranges. https://forums.aws.amazon.com/ann.jspa?annID=1528.

[13] ELB. http://aws.amazon.com/elasticloadbalancing.

[14] Heroku. http://heroku.com.

[15] Knock. http://code.google.com/p/knock.

[16] Passive DNS Network Mapper. http://code.google.com/p/dnsmap/.

[17] PlanetLab. http://planet-lab.org/.

[18] B. Agarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese. EndRE: An End-System Redundancy Elimination Service for Enterprises. In *NSDI*, pages 419–432, 2010.

[19] A. Akella, S. Seshan, and A. Shaikh. An Empirical Evaluation of Wide-area Internet Bottlenecks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 101–114. ACM, 2003.

[20] A. Akella, S. Seshan, and A. Shaikh. Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies. In *USENIX Annual Technical Conference, General Track*, pages 113–126, 2004.

[21] T. Benson, A. Akella, and D. A. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010.

[22] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu. A First Look at Inter-Data Center Traffic Characteristics via Yahoo! Datasets. In *INFOCOM, 2011 Proceedings IEEE*, pages 1620–1628. IEEE, 2011.

[23] I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras. Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 481–494. ACM, 2012.

[24] J. Edberg. Post-Mortem of October 22, 2012 AWS Degradation. http://techblog.netflix.com/2012/10/post-mortem-of-october-222012-aws.html.

[25] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift. More for Your Money: Exploiting Performance Heterogeneity in Public Clouds. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 20. ACM, 2012.

[26] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208. ACM, 2009.

[27] 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 *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 190–201. ACM, 2009.

[28] C. Labovitz. How Big is Amazon's Cloud? `http://deepfield.net/blog`, April 2012.

[29] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.

[30] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang. CloudProphet: Towards Application Performance Prediction in Cloud. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 426–427. ACM, 2011.

[31] R. McMillan. Amazon's Secretive Cloud Carries 1 Percent of the Internet. `http://www.wired.com/wiredenterprise/2012/04/amazon-cloud/`, 2012.

[32] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das. Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):34–41, 2010.

[33] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *USENIX Security Symposium (SSYM)*, 1998.

[34] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM, 2009.

[35] J. Schectman. Netflix Amazon Outage Shows Any Company Can Fail. `http://blogs.wsj.com/cio/2012/12/27/netflix-amazon-outage-shows-any-company-can-fail/`, 2012.

[36] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-potato Routing in IP Networks. In *ACM SIGMETRICS Performance Evaluation Review*, volume 32, pages 307–319. ACM, 2004.