A Distributed Architecture for the Monitoring of Clouds and CDNs: Applications to Amazon AWS

A Distributed Architecture for the Monitoring of Clouds and CDNs: Applications to Amazon AWS

Ignacio Bermudez, Stefano Traverso, Maurizio Munafò, Marco Mellia
DET, Politecnico di Torino, Italy – {lastname}@tlc.polito.it

***Abstract*—Clouds and CDNs are systems that tend to separate the content being requested by users from the physical servers capable of serving it. From the network point of view, monitoring and optimizing performance for the traffic they generate is a challenging task, given the same resource can be located in multiple places, which can in turn change at any time. The first step in understanding Cloud and CDN systems is thus the engineering of a monitoring platform. In this paper, we propose a novel solution which combines passive and active measurements, and whose workflow has been tailored to specifically characterize the traffic generated by Cloud and CDN infrastructures.**

We validate our platform by performing a longitudinal characterization of the very well known Cloud and CDN infrastructure provider Amazon Web Services (AWS). By observing the traffic generated by more than 50,000 Internet users of an Italian ISP, we explore the EC2, S3 and CloudFront AWS services, unveiling their infrastructure, the pervasiveness of web-services they host, and their traffic allocation policies as seen from our vantage points. Most importantly, we observe their evolution over a two-year long period.

The solution provided in this paper can be of interest for i) developers aiming at building measurement tools for Cloud Infrastructure Providers, ii) developers interested in failure and anomaly detection systems, and iii) third-party SLA certificators who can design systems to independently monitor performance. At last, we believe the results about AWS presented in this paper are interesting as they are among the first to unveil properties of AWS as seen from the operator point of view.

I. Introduction

Last years witnessed the growth of Cloud infrastructures that provide computing, storage and offloading capabilities on remote datacenters, offering the opportunity to companies to run their web-services at very competitive costs. Following the definitions provided in [1], the *Infrastructure Providers* (*IP*) offer the "iron", i.e., the hardware infrastructure, and *Infrastructure as a Service* (*IaaS*) products represent the virtualization technology needed to exploit such hardware. In other words, through virtualization, a large set of computing resources, such as storing and processing capacities can be split, assigned, and dynamically sized to satisfy customers' demand. Customers, or *tenants*, are companies that offer their *web-services* without carrying on costs and risks of building and managing their own physical infrastructure. Many popular companies like Dropbox and Netflix to name a few, successfully rely on IaaSes to provide their web-services.

Content Delivery Networks (CDNs) represent another critical service of the contemporary Internet and are responsible for moving around large shares of traffic served by thousands of datacenters to the end-users scattered worldwide [2]. A particular feature Clouds and CDNs have in common is that, from the network point of view, they both separate the web-service or content from the actual server serving it. This dramatically complicates the task of monitoring the web-services and the provided quality. Indeed, for instance, in the case of the CDN, the originating server offering a given content can delegate its distribution to surrogate servers, or caches. Similarly for Clouds, the separation between content and the server generating it is brought by the virtualization layer. In addition, virtual servers are often hidden by front-end servers, e.g., load balancers. This leads to a tangle that is very hard to understand.

Given the penetration of Cloud- and CDN-based web-services, it is crucial to continuously monitor these systems. In this paper we address the problem of shedding light on Cloud and CDN infrastructures. We provide guidelines to build a distributed measurement platform that leverages passive and active measurements, continuously or on-demand, at a wide variety of scales, and is specifically tailored to i) appraise the distributed infrastructure of Clouds and CDNs, and ii) assess and characterize the web-services relying on them.

Despite the prominent relevance of Clouds and CDNs in nowadays Internet scenario, practically no study has focused on the aspect of building a comprehensive measurement framework to monitor them from the perspective of an Internet Service Provider (ISP). The measurement platforms that have been proposed in the last years, such as PerfSONAR [3] and RIPE Atlas [4], are broad and not flexible enough to be adapted to monitor exclusively Clouds and CDNs. Other solutions such as CloudCmp [5] and C-Mart [6], focus on comparing the performance of different Cloud providers by running benchmarks. Cloud providers often offer APIs (e.g., Amazon CloudWatch[1]) to provide a detailed view on the Cloud's internal state (e.g., per-instance CPU utilization, disk read/write rates, etc.). The measurement platform we design in this paper complements existing monitoring approaches, adding a detailed user-side perspective on the network traffic generated by the web-services running on the Cloud or CDN. Moreover, differently from in-IaaS performance evaluation frameworks, our platform can answer other kinds of questions: *how much traffic does a Cloud/CDN generate? which kind of web-services does it run? how well do the web-services perform in terms of QoS? how geo-diverse is their deploy-*

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane") and from the BigData@Polito project.

[1] http://aws.amazon.com/cloudwatch/

ment? are there problems due to the network, to the datacenter, or to a particular virtual instance? how do Cloud and CDN infrastructures evolve over time? To answer those questions, one has to properly identify i) which web-services run on the targeted IaaS, ii) where the datacenters are, and iii) define proper measurement indexes.

Answering such kind of questions constitutes a big challenge, and measurements are the first step to shed light on the obscure functioning of Clouds and CDNs, and to open the possibility for a better design and management.

We instantiate the general measurement framework envisioned by the mPlane project [7], [8] to specifically monitor Cloud/CDN-based web-services. Thus, we first present how we adapt the mPlane components and workflow for this specific monitoring task. Second, we detail the metrics and the methodologies we adopted to evaluate the performance, the QoS and the network cost of web-services relying on Cloud and CDN systems. Third, we describe how the whole measurement process can be made automatic at every step, from the collection of passive measurement of traffic to the presentation of results.

We validate our measurement platform to characterize Amazon Web Services (AWS), the leading Cloud Infrastructure Provider. AWS includes a complete range of IaaS products. The most well-know are Elastic Compute Cloud (EC2)[2] which provides resizeable compute capacity in the Cloud by means of virtualized servers, and Simple Storage Service (S3)[3] which offers a service to store/retrieve files into/from the Cloud. Less known, AWS integrates a CDN, named CloudFront[4], to distribute content to end-users with low latency and high data transfer speeds. Our analysis relies on network traffic that has been collected from our University campus and from three large Points of Presence (PoPs) of an Italian country-wide ISP. In total we observed the traffic of more than 50,000 users for a two-year long period. We dig into various time portions of our dataset to analyze the evolution over time of AWS, to reveal, e.g., the number of datacenters[5], their locations, and performance as perceived by the ISP customers. Second, we investigate several popular Cloud/CDN-based web-services that run on AWS to show their dynamics and how they perform with respect to the overall performance of the datacenters they rely on.

A. Our Contributions

In a nutshell, the contribution of this paper can be resumed in the following:

• We provide the guidelines to design and engineer a platform to passively and actively monitor Cloud and CDN infrastructures in a continuous manner.

• We present and exploit some novel active measurement technique (e.g., *HTTP-knocking*) to overcome the separation between web-services and servers that is caused by virtualization (surrogate caches) in Cloud (CDN) systems. Identifying the physical servers which are hosting given web-services is indeed a crucial task to investigate the traffic generated by Clouds and CDNs.

• We validate such platform by running a comprehensive analysis of Amazon Web Services. Some of the main observations we obtain from the results of such analysis are i) some of AWS datacenters have notably improved their performance between 2012 and 2013, ii) most of the tenants tend to concentrate their web-services on a single datacenter, thus increasing the latency experienced by farther end-users, and possibly the cost for the network.

B. Possible Use-cases

The measurement platform for Cloud and CDN traffic which we propose in this paper may be useful for a plethora of IT figures: i) Researchers are interested in understanding how Clouds and CDNs work, how they perform, and how their design can be improved. ii) Tenants need a (preferentially third-party) assessment of performance to make informed decisions when choosing the best IaaS on the market to roll out their web-service; Once deployed, they need to verify the Service Level Agreement (SLA) they subscribed to, possibly from the user-side perspective. E.g., they aim at evaluating the reactiveness and reliability of virtual instances hosting their web-services, or the goodput of the CDN caches serving their contents. Moreover, they need tools to accelerate the QoS troubleshooting process, and to capture the root cause behind a QoS degradation (which may also reside out of the Cloud/CDN, i.e., in the network or in the client). iii) ISPs can exploit measurements to optimize their networks for specific web-services their customer are interested in, and even offer a third-party cross-Cloud-provider monitoring service to tenants interested at understanding how their web-services perform from the point of view of the end-users. iv) Finally, end-users can be offered automatic failure and anomaly detection systems which so far rely on communities' reports [9] or active measurements [10].

The rest of this paper is organized as follows: Sec. II presents the measurement architecture and its components. In Secs. III, IV-A and IV-B we describe the methodology, the techniques and the metrics, respectively, we design specifically for the analysis of Cloud and CDN traffic. Sec. IV-C overviews the data collection procedure and the datasets we employ for this study. In Sec. V we present the measurement results about the AWS infrastructure, and the web-services running on it. Sec. VI presents the performance results. Finally, Sec. VII summaries the related work, and Sec. VIII concludes the paper.

II. Measurement Architecture

In this section we first provide a high level overview of the distributed measurement architecture proposed by the mPlane project (more details on the generic infrastructure are available in [7], [8]). Then, we describe how we implement and deploy mPlane components for this study.

[2] http://aws.amazon.com/ec2/

[3] http://aws.amazon.com/s3/

[4] http://aws.amazon.com/cloudfront/

[5] Datacenters are named *Availability Zones* in AWS terminology. We use both terms interchangeably in the remainder of the paper.

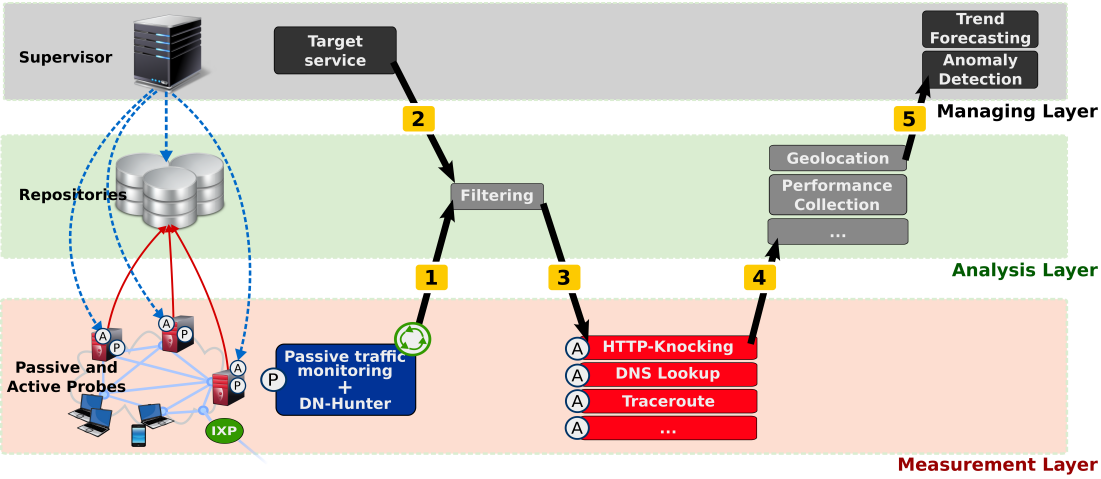


Figure 1: The mPlane measurement architecture and its components (left-hand side of the figure): probes continuously collect traffic measurements, which are periodically exported to the repository (red arrows). The repository consolidate, filter and aggregate the measurements for further analysis. The Supervisor orchestrates other components, instruments the probes to launch active measurements and the repository to launch further data processing (dotted blue arrows). The right-hand side of the figure reports the monitoring workflow (black arrows refer to interactions among different components): 1) Passive probes collect traffic measurements and export them to the repository, 2) the Supervisor instruments the repository to filter the Cloud/CDN traffic we are interested into, 3) active measurements are run to augment the information collected by passive probes, 4) aggregate measures, performance indexes, location of the servers, etc. are executed, and 5) the Supervisor launches sophisticated algorithms with different high level aims.

A. mPlane Measurement architecture

As depicted in the left part of Fig. 1, the architecture consists of three kinds of components, namely *probes*, *repositories* and *supervisors*. They interoperate thanks to a standard protocol, and are logically organized into three main layers:

1) **The Measurement Layer:** it consists of a set of probes located at several vantage points within the monitored network. In general, they generate large amounts of data. The measurement may be *active*, e.g., the output of a simple *traceroute*, or may be *passive*, e.g., the packet level trace of traffic flowing on a link. Each probe performs measurements either on demand or continuously, and then periodically exports data to curb local storage space requirements.
2) **The Analysis Layer:** it consists of *repositories* which collect and aggregate data generated by probes. Apart from the storage capacity, the Analysis layer includes analysis routines which iteratively process the measurement data. Such processing may involve filtering, classification and computation of performance indices. The result is a higher level of aggregated measurements that can form another source of data as well.
3) **The Management Layer:** it is governed by the *Supervisor* which, beside orchestrating other components, it further processes the aggregated data made available by the Analysis Layer. Here, high level algorithms are executed, for instance, for anomaly detection or trend estimation tasks.

B. mPlane Deployment

1) **Probes:** For this study, we rely on a measurement layer composed by three probes installed in the operational network of an Italian nation-wide ISP. In total our probes monitor the traffic of more than 50,000 residential users regularly accessing the Internet through ADSL and FTTH technologies. A fourth probe is installed in Politecnico's campus network to monitor the traffic of 12,000 users, including students and personnel accessing the Internet through wired and WiFi networks. In our deployment, the passive probes are based on Tstat[6], an advanced passive monitoring tool developed at Politecnico di Torino. It analyses packets sent and received by terminals inside the monitored network [11]. Tstat rebuilds TCP and UDP flows carried on the monitored link. For each, it logs more that 100 statistics, e.g., IP addresses and port numbers, number of bytes, protocol used, etc. Tstat generates one-hour long logs, temporarily stored on the probe. Tstat does not log any personal information about end-users to preserve their privacy. We equip the same machines running Tstat to act as active probes on demand, for instance, to run *traceroute* and *HTTP-knocking* tests (see Sec. III for a detailed description).
2) **Repository:** Our repository consists of a NAS device with a 32TB storage capacity. It periodically downloads data from the four probes at night, i.e., during low network usage period, in order to not interfere with Tstat activity. Once on the repository, we consolidate passive and active measurements in tables, and a set of automatic scripts processes them to filter out useless information,

[6]http://www.tstat.polito.it

compute index distributions and produce aggregated time-series.

3) **Supervisor:** since we do not employ any advanced analysis for this study, the Supervisor's main task consists of instrumenting the probes to run passive and active measurements, and the repository to periodically download the measurement data.

III. Analysis Methodology for Cloud and CDN traffic

In our design, we follow the principle of preferentially exploiting passive measurements, and limiting active measurements as much as possible. The major benefit of exploiting passive measurements is that they not alter the working point of the system which faces only the actual workload generated by users. As such, they allow us to characterize the actual performance of web-services being accessed by regular customers.

A. Background

We have to face several challenges when passively monitoring Clouds and CDNs: the separation between *web-service* (or *content*) and *server*, and the wide variety of applications make it hard to identify which web-service end-users access, and in which datacenter the server hosting the web-service resides. In Clouds and CDNs indeed, virtualization, load-balancing and migration techniques do not allow to simply rely on information collected from the Network Layer to identify the web-service, i.e., the server IP address and the name of the Organization that officially registered it often give very limited information on what is being served. For instance, different web-services can share the same IP address, e.g., as done by CDNs. In addition, the increasing adoption of TLS/SSL encryption by content providers [2] makes the classic Deep Packet Inspection (DPI) approach ineffective.

Consider the same server hosting *www.acmegame.com* and *www.acmeshop.com*; both are delivered only via HTTPS, and run on the *AcmeCloud* infrastructure. Via passive monitoring, the only useful information that can be obtained is related to the IP addresses of instances hosting them; supposing IP addresses are officially registered to *AcmeCloud*, we can discover that both are handled by *AcmeCloud*. However, we can not infer any indication about the web-service just by looking at information from the Network and Transport Layers.

Similarly, identifying which datacenter is being used, and its location is not trivial. This information is important to understand eventual performance issues, e.g., related to high latency due to the distance between the user and the datacenter. Indeed, it is well-known that most of publicly available geolocation databases are not reliable when querying information of Cloud or CDN providers [12]. Furthermore, this information becomes often outdated given the constant evolution of Cloud infrastructures.

All these observations make the monitoring of traffic generated by Cloud and CDN systems a complicated task. In the following we describe the techniques we engineer to overcome such hurdles, and how we embed them to work in the measurement platform described in Sec. II-B.

B. The Measurement Workflow

In Fig. 1 we depict the steps which compose our monitoring workflow. Each operation is mapped to the proper measurement component.

Step 1 - Passive monitoring (Tstat + DN-Hunter): Passive Tstat probes provide the continuous collection of traffic logs. This data constitutes the basic brick for our analysis. To identify the content retrieved by end-users from a server belonging to a certain organization, we exploit a Tstat extension, named DN-Hunter [13]. It snoops DNS responses obtained by clients when resolving hostnames into IP addresses, and then uses this information to annotate TCP/UDP flows with the the original server hostname. Following the example described above, end-user's client accessing *www.acmegame.com* has first to resolve the server hostname into a list of IP addresses by contacting the DNS resolver. Then, the client contacts one of the returned IP addresses to fetch the actual content. DN-Hunter records all DNS responses, and then it associates the server hostname to the TCP flows originated by the same client and directed to one IP address of servers returned in a given response, i.e., associating *www.acmegame.com* to the observed TCP flow. This key feature allows us to gather information about the original web-service name contacted by clients. Details on DN-Hunter are out of the scope of this paper, and can be found in [13].

Step 2 - Target service filtering: The definition of the set of Clouds or CDNs to consider for the analysis is given by the analyst through the Supervisor. This set is used to filter the raw data. In our validation, we preserve only those flows whose server IP addresses are registered to Amazon according to the MaxMind database[7]. The output is a subset of Tstat logs, where only traffic related to the targeted organizations is present.

Step 3 - Augmenting information by active probes: We next augment the information about servers and infrastructure by means of active measurements. This operation is executed by active probes, marked with "A" in Fig.1, installed within the monitored network to guarantee that no bias is introduced into the results. In parallel, and for each discovered server IP address, active probes i) execute the *HTTP-knocking*[8], ii) perform a DNS reverse lookup to retrieve the "Type-A" record [14], i.e., the original name of IP address as registered by the targeted organization, and iii) run *traceroute* to retrieve network path information, such as number of traversed hops or autonomous systems. We can also plug extra modules to retrieve additional information.

Step 4 - Geolocation and performance collection: The repository runs a geolocation routine to identify the server's physical location. The geolocation of IP addresses is a well-known problem. Common public databases, such as RIPE [4], ARIN [15] or MaxMind, are not reliable when seeking for information about the physical location of CDN edge servers and

[7] http://www.maxmind.com//en/organization

[8] Probes perform a HTTP HEAD requests to server. The `Server` field of the HTTP response is inspected to classify the hosted IaaS product. For instance, `AmazonS3` or `CloudFront` is returned for AWS S3 and CloudFront servers, respectively.

Cloud datacenters [12], and we must adopt other methodologies. In our validation, we rely on the information provided at DNS level again. Indeed, the record returned when performing the reverse lookup of IP addresses unveils information about server placement by exposing the International Air Transport Association (IATA) airport location identifiers. This is common with other CDN infrastructures [16]. Alternatively, we can instrument active probes to measure "network" distances, which combined with multilateration techniques can lead to fairly precise geolocation [17].

Once we collect all above information, we compute performance measurements on different dataset portions and at several time scales. Details are provided in Sec.IV.

Step 5 - Visualization and time analysis: At last, the repository exports the results of its processing to the supervisor, which processes the time series, and presents them in a dashboard for an easy visualization. The same results can be employed as input for high-level algorithms such as change detection, or trend comparison and forecasting.

In a nutshell, passive probes continuously provide new data (step 1), which we filter at the repositories (step 2) as soon as it arrives. We run active measurements (step 3) and geolocation (steps 4) asynchronously, e.g., once per day. The extraction of performance indexes is executed typically following a batch approach, i.e., every $\Delta T = 5$min a new batch is formed and analyzed. The final output is thus a time series of aggregated measurements, which we store in the repository again, and sophisticated algorithms can further process for high level purposes.

IV. Measurement and Dataset definition

In the following section we describe the metrics and datasets we consider to characterize Clouds and CDNs.

A. Per-flow Metrics

Among the measurements provided by Tstat, for each flow, we consider i) the server IP address, ii) the hostname as retrieved by DN-Hunter, iii) the flow RTT, iv) the amount of bytes (sent and received), v) the application layer protocol, e.g., HTTP and HTTPS, and vi) timestamps of packets that are instrumental to obtain further indices – see Fig. 2. These metrics are straightforward to monitor, and details can be found on [11]. In particular, we define:

1) Response Time: it is the time the server employs to send the reply after receiving the first request from a client. Let T_{Ack} be the timestamp of the first TCP ACK message server sends with the relative ACK number greater than 1, i.e., acknowledging the reception of some data sent by the client. Let $T_{Response}$ be the timestamp of the first TCP segment the server sends carrying application data. The define the response time[9] as

$$\widehat{\Delta R} = T_{Response} - T_{Ack}. \quad (1)$$

[9]Notice that the probe measures the timestamps at vantage points close to the users. Therefore, for some metric X (as RTT_{CS}) we can only gauge its estimated measure $\hat{X}$.

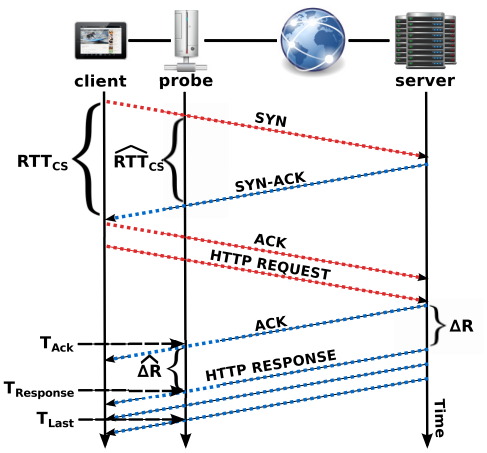


Figure 2: Example of time line of a TCP-based client-server transaction.

As depicted in Fig. 2, for HTTP flows, it represents a measure of the time the server takes to elaborate and transmit the response for the first HTTP request[10] (e.g. an HTTP response).

2) Flow Goodput: we define it as the rate at which the server delivers information generated at Application Layer to the client. Let $T_{Response}$ and T_{Last} be the timestamps of the first and the last data packet the server sends, and let D be the size of the application payload the server sends. We thus define the server goodput as

$$G = \frac{D}{T_{Last} - T_{Response}}. \quad (2)$$

To avoid the bias of short-lived flows, we evaluate the server goodput only on flows for which $D > 500kB$.

B. Network Cost

We aim at evaluating the cost sustained by the network to transport data generated by AWS servers to the end-users. To this extent, we define the Network Cost as the weighted average of the distance traveled by information units. Formally, given a TCP flow from client c to server s, let $B(c,s)$ be the amount of exchanged bytes, and let $D(c,s)$ be the distance between client c and server s. We compute the resulting network cost $\beta(s)$ for a given server s as

$$\beta(s) = \frac{\sum_c D(c,s)B(c,s)}{\sum_c B(c,s)}. \quad (3)$$

The average network cost of servers in datacenter $\mathcal{S}$ results

$$\beta(\mathcal{S}) = E[\beta(s)|s \in \mathcal{S}]. \quad (4)$$

We can thus consider different definitions of distance, $D(c,s)$: the TCP connection average RTT[11], the number of traversed

[10]We consider the response time for HTTP flows whose request is contained in one TCP segment to avoid the bias introduced by multiple segment requests.

[11]Observe that installing the probes in PoPs on the path between user's access networks and the servers allows us to gauge estimations of the round trip time $\widehat{RTT}_{CS}$ and number of hops $\widehat{AS}_{CS}$ that exclude the eventual noise introduced by the access network (see Fig 2). For instance, we filter out extra delays introduced by lossy WiFi or ADSL channels.

AS or hops on the path[12], the geodetic physical distance. These are leading respectively to $D^{RTT}(c,s)$, $D^{AS}(c,s)$, $D^{hops}(c,s)$, $D^{km}(c,s)$. Thus, we obtain different network cost metrics β^{RTT}, β^{AS}, β^{km}, respectively. Observe, that β^{RTT} and β^{km} represent good indices to understand how large is the distance packets have to travel before reaching their destination. As the RTT linearly increases with the physical distance between clients and servers [18], we expect them to be tightly related. Instead, β^{AS} allows us to roughly understand how large is (in terms of hops) the distance traveled before reaching the Cloud provider's network (AWS in our case), or conversely, how many Autonomous Systems the content has to pass through before it is delivered.

For the sake of simplicity, we employed β^{RTT} and β^{AS} in our characterization of Amazon AWS.

C. Datasets

For this study, we rely on a measurement layer composed by three passive probes installed in the operational network of an Italian nation-wide ISP. One further probe was installed in Politecnico's campus network. In the following we present a study of a snapshot of the measurements that form a "static" dataset.

Our dataset consists of traffic measurements which we collected since March 2012 to September 2013, observing end-users normally accessing the Internet. In the following, we restrict our analysis on traffic collected during two entire weeks: 1st to 7th April 2012 (*1Week-2012*), and 10th to 16th June 2013 (*1Week-2013*). We complement these one week long traces with a longer one, namely *3Months-2013*, which spans a period of three months, from April to June 2013 during which we consider one day for each week. Table I resumes the details about considered traces.

Trace	Period	Observed Days	Connections
1Week-2012	01/04/12-07/04/12	7 days	7.02 M
1Week-2013	10/06/13-16/06/13	7 days	12.2 M
3Months-2013	01/04/13-26/06/13	12 days	21.6 M

Table I: Measurement traces considered in this study.

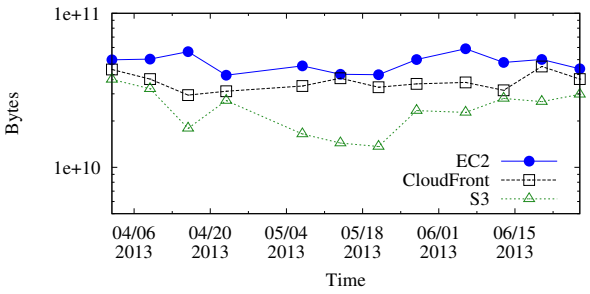
In order to minimize the bias due to changes in the measurement scenario, we present results collected from the same vantage point, i.e., by one of our three ISP probes where more than 16,000 users are present. Results from other vantage points show identical characteristics so that the findings we present are general and not biased by the observation point.

We acknowledge that some of the results in this paper are the point of view of a single ISP. Naturally, we expect that some observations can change if we analyze traffic in another ISP or geographical region.

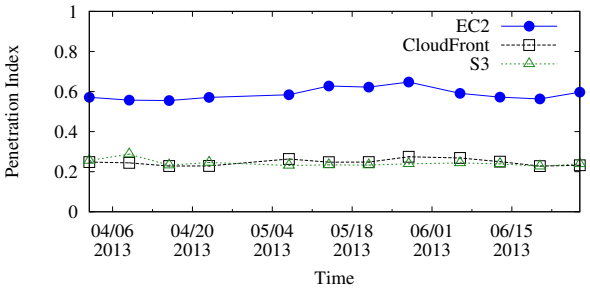
V. Characterization of AWS

As explained in Sec. III, we parse the server hostname and the HTTP response to a HTTP-knocking test to extract information about the IaaS product and geolocation of servers.

[12]The number of traversed AS is obtained running a *traceroute* from the probes and checking the AS of returned routers. For the number of hops we simply count the number of traversed routers.



(a) Exchanged data volume.



(b) Penetration index.

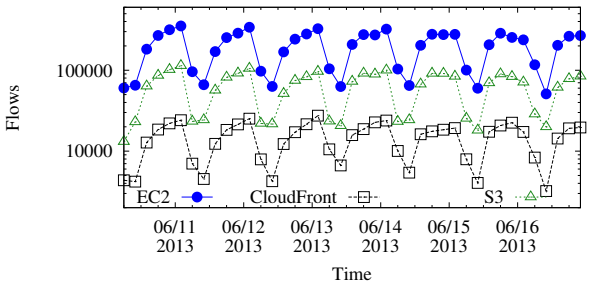
Figure 3: Evolution over time of different diffusion indexes for EC2, S3 and CloudFront services in *3Months-2013* trace.

Our measurements show that EC2, CloudFront and S3 servers use different ranges of IP addresses, thus easing their identification. Furthermore IP addresses are statically assigned for S3 and CloudFront servers. As such, we can easily split traffic among EC2, S3 and CloudFront.

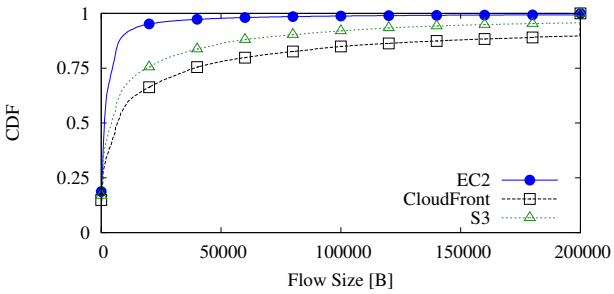
Considering server geolocation, the standard International Air Transport Association (IATA) airport location identifiers are used in the Type-A records of AWS servers, e.g., for CloudFront the Type-A name is in the form *server-a-b-c-d.**AIR**.r.cloudfront.net*, where AIR is the IATA airport code of the nearest large airport. We verify their accuracy with the multilateration technique described in [17]. This notably simplifies the geolocation operation in our processing. In the remainder of this paper, we use IATA codes to identify datacenters instead of conventional names of AWS Availability Zones.

Next, we present a high level characterization of the traffic and workload AWS generate. For each day in the *3Months-2013* trace, we compute the volumes of data (in bytes) exchanged with AWS servers and the fraction of customers that contacted at least one AWS service, i.e., the penetration index of AWS.

Plots of Fig. 3 report our results. Interestingly, both plots of Fig. 3 show that the footprint of AWS is rather stable in time, indicating that the growth rate of EC2, S3 and CloudFront is quite marginal for the considered period: On a daily basis, about 60% of end-users access some EC2 hosted web-service, exchanging about 50GB of traffic. S3 comes second, with approximatively 25% of end-users accessing directly to S3 services, and exchanging a volume of about 35GB per day. Scaling this to the total number of ISP customers, EC2 and S3 generate about 10TB of daily traffic, corresponding to about

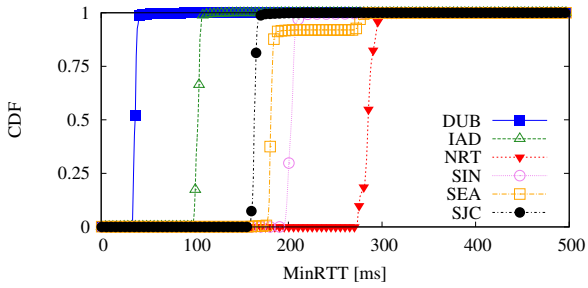


(a) Evolution over time of the number of flows.

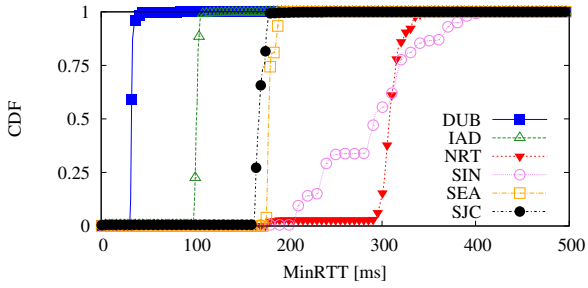


(b) Distribution of the flow size.

Figure 4: Flow information for for EC2, S3 and CloudFront services. *1Week-2013* trace.



(a) 2012



(b) 2013

Figure 5: Distributions of the minimum RTT measured in our EC2 datasets for different datacenters.

3% of total web traffic.

Even if we expect CloudFront and S3 to handle large files, EC2 is the service with the largest footprint. To better show this, top plot in Fig. 4a presents the evolution over time of the number of flows for EC2, S3 and CloudFront. Fig. 4b details instead the Cumulative Distribution Function (CDF) of the flow size. As shown, the number of flows handled by S3 and CloudFront are one and two orders of magnitude lower that EC2, but they carry larger payload than EC2 (the average amount of data are 39kB, 257kB and 383kB for EC2, S3 and CloudFront, respectively).

Further, around 50% of CloudFront flows present a size smaller than 10kB, mostly probably being CSS or JavaScript files employed for the rendering of web-pages. 20% of flows, which present a size larger than 100kB, carry binary data, e.g., Flash objects or images. Differences between size of contents served by different caches are negligible. S3 flows show in general the same size distribution as CloudFront flows. The average size of contents distributed by CloudFront and S3 is 78kB for both. EC2 flows are smaller in general, being the 60% of them less than 1kB. These could carry small XML files or JSON messages directed to APIs. For these files, the TCP three-way-handshake and tear-down procedures last longer than the data transfer.

Summary 1: *CloudFront and S3 host contents which present a larger size with respect to EC2. However, given the popularity of web-services relying on EC2, the overall amount of traffic EC2 generates is larger than S3 and CloudFront.*

We now focus on the spatial allocation of traffic and resources among different AWS datacenters. We detail the traffic they generate, the number of web-services they host, and the costs for the network. We consider both *1Week-2012* and *1Week-2013* traces.

A. EC2 and S3

Tables II and III report information about EC2 and S3, in 2012 and 2013, respectively. Each row in the table represents traffic exchanged with a different datacenter, that we identify with its IATA code. For each of them, we detail the number of observed server IP addresses, the shares of traffic, the share of hosted web-services[13], the network costs β^{RTT} and β^{AS}. To ease the visualization we highlight the most significant information using bold fonts.

Several observations hold. First, the number of server IP addresses for EC2 is much larger than S3 and CloudFront. This is due to the nature of EC2 itself, that, allocates indipentend EC2 instances, each one, in general, equipped with a different public IP address. For S3 the same server can deliver different contents by using different URIs. Allocating multiple IP addresses is thus useless in this case.

Second, the datacenters located in Virginia (IAD) and Ireland (DUB) are the most used from the perspective of an ISP located in Italy. We highlight in bold font the most notable statistics about these two datacenters. California (SJC) comes third, handling about 8% and 7% of EC2 web-services in 2012 and 2013, but only less than 2% of traffic.

The number of server IP addresses we observe in 2012 (for both EC2 and S3) and 2013 highlights that AWS resources have doubled in one year. Clearly, this has performance implications, as we will show later.

[13]We identify a web-service with the domain name of the hosting server returned by DN-Hunter.

ID	#IPs		Exchanged Data		Web-services	β^{RTT} [ms]		β^{AS}
	EC2	S3	EC2	S3	EC2	EC2	S3	
IAD	**6603**	100	**86.03%**	**65.57%**	**68.95%**	**113.97**	**116.18**	3
DUB	**1218**	11	**12.29%**	**32.86%**	**20.21%**	**48.73**	**43.77**	3
SJC	620	7	1.60%	–	7.94%	182.14	174.81	4
NAR	20	0	–	–	0.47%	–	–	4
SIN	75	0	0.03%	–	1.62%	228.10	–	3
SEA	0	0	–	–	0.81%	–	–	4

Table II: Amazon's datacenters hosting EC2 and S3 we located in *1Week-2012*.

ID	#IPs		Exchanged Data		Web-services	β^{RTT} [ms]		β^{AS}
	EC2	S3	EC2	S3	EC2	EC2	S3	
IAD	**12697**	199	**87.71%**	**15.28%**	**59.61%**	106.52	118.19	**3**
DUB	**4042**	39	**10.20%**	**81.89%**	**26.79%**	38.14	44.83	3
SJC	984	24	0.78%	1.60%	5.96%	172.29	184.81	3
NAR	361	0	–	–	1.44%	318.16	–	3
SIN	188	18	–	–	1.26%	297.18	303.11	3
SEA	768	0	1.0%	–	4.95%	184.81	–	3

Table III: Amazon's datacenters hosting EC2 and S3 we located in *1Week-2013*.

Interestingly, both the large unbalance in the number of instances (number of IP addresses in EC2 column) and the share of web-services shows that the IAD datacenter is by far the most popular among tenants running EC2 instances: IAD generates more than 85% of the total amount of EC2 traffic, i.e., roughly 7 times more than the volume handled by the DUB datacenter. This confirms that IAD datacenter is much larger than all the others.[14]

Consider now the CDF of the minimum RTT among each flow depicted in Figs. 5a and 5b for 2012 and 2013, respectively. The minimum RTT CDF is very sharp, reflecting the network distance from the vantage point to the datacenter.

As expected, DUB is the closest for European users (Dublin, IR, $RTT < 30$ms), followed by IAD (Dulles, VA, $RTT > 100$ms), US West-Coast datacenters in SJC (San Jose, CA, $RTT > 160$ms) and SEA (Seattle, WA, $RTT > 170$ms). Asian datacenters have very high RTT, with NRT (Narita, JP) being close to 300ms. Curiously, the minimum RTT towards SIN (Singapore, SG) datacenter is affected by some very heterogeneous multipath routing policy which we do not observe in 2012, but is clearly evident in 2013. We double-check this against *traceroute* data and find it to be still present at the time of the writing of this paper.

Clearly, the RTT plays important role when considering application performance and Quality of Experience (QoE). From a network point of view, the cost of transferring data is also important. β^{RTT} cost, indeed, looks sizeable for IAD, being from 230% to 490% more expensive than the DUB datacenter. One may expect DUB to be the best candidate to host EC2 instances for serving Italian (and European) end-users. Our intuition is that, for the sake of a simpler management, tenants are oriented to deploy their web-services on one datacenter only. IAD represents their first choice because of its lower price.[15]

This bias is also explained by the lack of location-aware load-balancing policy for EC2. Indeed, AWS offer load-balancing-based forwarders for incoming traffic, but none of these are based on the geolocation of end-users. Thus load-balancing is effective among servers inside a give zone. This comes at the expenses of network cost, and, possibly, user experience.

Comparing 2012 with 2013, we observe no substantial change in shares for EC2. However, a huge shift has happened for S3 services: DUB generates more than 81% of traffic in 2013, while it was below 33% in 2012. Indeed, tenants can now exploit S3 load-balancers to route end-users' requests towards the closest datacenter, thus improving the latencies and saving bandwidth.[16]

Focusing now on β^{AS}, we see that no major differences are present when routing traffic to different datacenters. This reflects the fact that Amazon (and the monitored ISP) had very good peering agreements with many providers in 2012, and possibly improved them further in 2013.

At last, left plots in Figs. 6 and 7 report the evolution over time of the share of volumes of data traffic seen from the top three datacenters for EC2, for 2012 and 2013 datasets respectively. Each point refers to a 4h long time interval, covering an entire week. Other datasets and periods of time show very similar trends.

Starting from the leftmost plot, we show that EC2 IAD zone consistently provides much larger amount of traffic with respect to DUB and SJC. The gap between IAD and DUB has even widened in 2013 with respect to 2012, confirming that IAD datacenter is the most employed by tenants.

Center plot refers to S3. The today preference for the DUB datacenter is confirmed.

Summary 2: *We first observe that Amazon has notably increased the resources for the datacenters hosting EC2 and S3 between 2012 and 2013. Second, from a European ISP perspective, IAD and DUB datacenters are the two most employed by Amazon to offer EC2 and S3; however, we observe that there is a large unbalance between IAD and DUB, as the majority of EC2 and S3 web-services were hosted by IAD in 2012. Interestingly, in 2013 DUB has become the leading datacenter for S3 traffic in 2013. At last, in 2013, Amazon has improved its peering agreements with many providers, thus reducing the number of ASes traversed by its traffic. Interestingly, the authors of [19] lead to similar observations, but from a different country (US).*

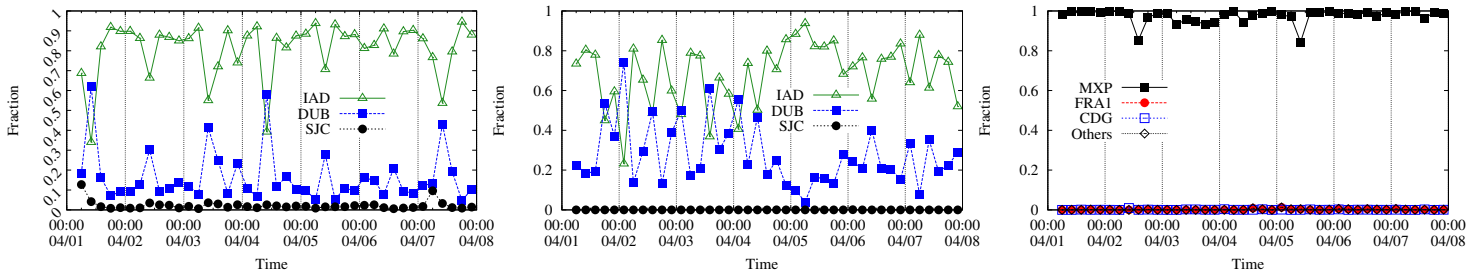
B. EC2 Web-service Analysis

Table IVa and Table IVb report the most popular web-services EC2 hosts, considering the number of flows and exchanged volume, respectively. In general, the type of web-service is quite heterogeneous. We observe that social games (Zynga, Farmville, Wooga and PlayFish), and by advertising companies (InviteMedia, 360yield) generated the largest portion of flows in 2012. In 2013, most of social gaming web-services such as PlayFish and Zynga severely downscaled their activity. This reflects the popularity of web-services among (Italian) end-users. The only online social gaming web-service

[14] http://aws.amazon.com/about-aws/globalinfrastructure/

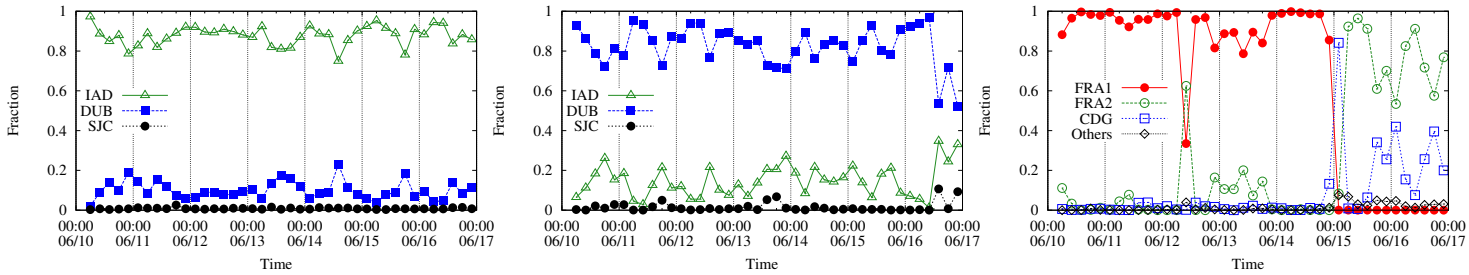
[15] http://aws.amazon.com/ec2/pricing/

[16] Tenants can upload S3 files to multiple Availability Zones, and Amazon provides load balancing automatically among them.



(a) EC2. (b) S3. (c) CloudFront.

Figure 6: Evolution over time of data traffic volume for the three considered AWS services. 2012 dataset.



(a) EC2. (b) S3. (c) CloudFront.

Figure 7: Evolution over time of data traffic volume for the three considered AWS services. 2013 dataset.

	IAD				DUB			
2012 dataset	Web-service	Flows%	DCs	IPs	Web-service	Flows%	DCs	IPs
	zynga	**13.98**	1	119	**wooga**	**24.17**	1	7
	farmville	**13.16**	1	86	**invitemedia**	**17.53**	3	73
	playfish	**9.43**	1	53	tp-cdn	6.21	2	5
	widdit	5.87	1	16	**360yield**	5.51	2	17
	amazonaws	4.05	6	8577	mydlink	2.24	1	6
	textsrv	3.25	1	10	scee	2.14	1	7
	chartbeat	3.23	1	33	cedexis-radar	2.11	4	5
	bidsystem	2.08	1	19	amazonaws	2.10	6	8577
	IAD				DUB			
2013 dataset	Web-service	Flows%	DCs	IPs	Web-service	Flows%	DCs	IPs
	trusted-serving	6.62	1	36	wind	5.40	1	2
	trkjmp	4.34	1	30	**360yield**	5.19	**2**	34
	chartbeat	3.83	1	57	**wooga**	3.37	**2**	50
	betrad	3.14	1	22	tp-cdn	3.26	1	14
	minecraft	2.53	1	60	**samsungosp**	3.14	**4**	304
	amazonaws	2.43	8	19074	tidaltv	2.51	2	27
	origin	2.41	1	249	pubnub	2.24	2	26
	instagram	2.19	1	71	up2potential	1.87	1	2

(a) By number of flows.

	IAD				DUB			
2012 dataset	Web-service	Volume%	DCs	IPs	Web-service	Volume%	DCs	IPs
	dropbox	**51.66**	**1**	625	**wetransfer**	**22.78**	**1**	16
	fruitninjafrenzygame	16.83	1	26	lastampa	12.79	1	1
	zynga	4.98	2	124	socialtrivia	7.11	2	3
	farmville	3.86	1	86	pagestreamer	4.66	1	1
	playfish	3.50	1	53	twww	3.31	1	12
	tapjoyads	0.97	1	96	wooga	2.52	1	7
	realtimetv	0.58	1	1	inter	2.47	2	3
	widdit	0.53	1	16	mo	2.45	2	7
	IAD				DUB			
2013 dataset	Web-service	Volume%	DCs	IPs	Web-service	Volume%	DCs	IPs
	dropbox	**63.76**	**2**	1308	**wetransfer**	**33.22**	**2**	30
	dropboxusercontent	**19.71**	**1**	91	lastampa	4.88	1	4
	haydaygame	1.28	1	20	samsungosp	3.85	4	304
	instagram	0.96	1	122	euronics	3.36	1	1
	clashofclans	0.37	1	20	wind	3.15	1	2
	tapjoyads	0.32	1	96	leroymerlin	2.57	1	4
	eamobile	0.27	1	82	jw	1.53	1	3
	trusted-serving	0.27	1	36	qpic	1.37	1	4

(b) By volume.

Table IV: The top contents hosted by EC2.

which is present in a leading position in both 2012 and 2013 datasets is Wooga.

Most of web-services generate traffic exclusively from a single datacenter, suggesting that the tenants do not replicate them among different Availability Zones, as also shown in [19]. The only notable web-services which rely on multiple datacenters are InviteMedia and 360yield.

Considering the top web-services by volume, we notice the important presence of storage web-services like Dropbox and WeTransfer. The former generates almost 80% of the EC2 traffic alone in 2013 dataset. Both Dropbox and WeTransfer doubled the number of IP addresses in 2013. We provide further considerations about the performance of some of the popular web-services in Sec. VI-C.

S3 web-services present similar properties as EC2. S3 web-services are in general for storage, advertisement and social games, but none of them appears to be accessed over different Availability Zones.

Summary 3: *We observe that i) in general tenants tend to rely on a single datacenter for the deployment of their web-services, and ii) storage web-services as Dropbox and WeTransfer are responsible for generating the largest share of traffic among EC2 web-services. These results confirm the observations in [19].*

C. CloudFront

Let us focus on CloudFront results reported in Tables V and VI. They detail statistics about the top 14 CloudFront caches observed in our datasets. To ease the visualization we highlight the most significant detected caches using bold fonts. Observe how biased was the preference towards the MXP (Milan) cache

	ID	#IPs	Exchanged Data (%)	β^{RTT} [ms]	β^{AS}
Caches	**MXP**	232	**98.03%**	**21.26**	**3**
	SFO	253	0.83%	175.21	4
	ANR	151	0.56%	41.48	3
	LHR	182	0.17%	31.60	3
	FRA	245	0.17%	21.87	2
	DUB	222	0.05%	49.76	3
	CDG	246	0.13%	38.43	3
	AMS	205	0.04%	29.88	3
	EWR	208	$< 0.01\%$	109.53	3
	NRT	115	$< 0.01\%$	–	4
	SEA	64	$< 0.01\%$	–	4
	SIN	51	$< 0.01\%$	–	3
	IAD	2	$< 0.01\%$	102.75	3
	SJC	–	$< 0.01\%$	–	4

Table V: Summary of Amazon's top 14 CloudFront caches we located in *1Week-2012*.

	ID	#IPs	Exchanged Data (%)	β^{RTT} [ms]	β^{AS}
Caches	**FRA**	**797**	**87.76%**	17.40	**3**
	CDG	597	**11.04%**	24.50	**3**
	AMS	755	0.84%	23.03	3
	LHR	653	0.42%	36.22	3
	NRT	453	$< 0.01\%$	309.56	3
	JFK	371	$< 0.01\%$	109.54	4
	DFW	315	$< 0.01\%$	153.81	3
	LAX	293	$< 0.01\%$	193.93	3
	SIN	291	$< 0.01\%$	333.34	3
	SFO	255	$< 0.01\%$	170.82	3
	HKG	240	$< 0.01\%$	330.36	3
	IAD	216	$< 0.01\%$	107.86	3
	JAX	164	$< 0.01\%$	147.02	3
	MXP	43	$< \mathbf{0.01\%}$	**41.48**	**3**

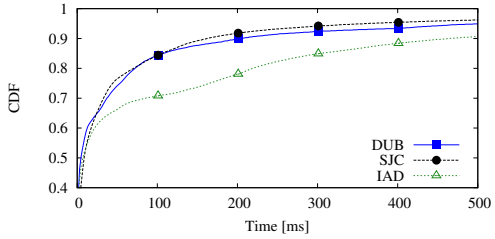
Table VI: Summary of Amazon's top 14 CloudFront caches we located in *1Week-2013*.

in 2012, which results the best cache considering any definition of network costs. Surprisingly, 2013 witnesses a significant change: the most employed caches are in Frankfurt (FRA) and in Paris (CDG). However, even if the geographic distance between ISP's end-users and the cache is larger for FRA than MXP, the β^{RTT} has decreased. Finally, focusing on the β^{AS} column, the number of AS to traverse to reach a CloudFront cache has on average decreased in 2013. This hints to some optimization on the CloudFront CDN policy.

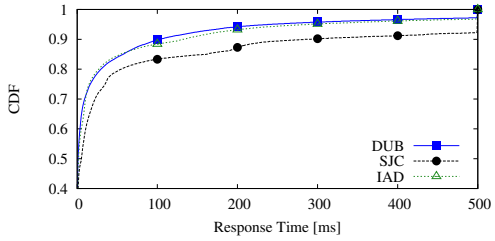
To complement these observations, we instrument active probes to run experiments in which we resolve 100 different web-service names hosted by CloudFront using more than 2,000 open DNS resolvers scattered worldwide. This process allows us to identify possibly different CloudFront caches. We find a total of 33 caches, each hosting a /24 subnet. We observe the CloudFront is effective in directing ISP end-users to the closest cache (at least in terms of RTT), as expected for a CDN. Indeed, only less than 2% of traffic is delivered from caches far away from end-users' position. This may be because of some end-users employing alternative DNS servers, different from those provided by their ISP. For instance, both OpenDNS and Google DNS servers direct requests from the ISP end-users to farther caches. This is consistent with findings in [20].

Fig. 7c depicts a further difference between *1Week-2013* and *1Week-2012*. The plot shows that the load distribution is not strongly polarized towards one cache as in Fig. 6c where the preference for the MXP cache is very strong. Fig. 7c shows that cache FRA1 handles a large fraction of CloudFront traffic until the midnight of June 14, but a drastic shift appears since June 15, when FRA1 apparently disappears leaving space to FRA2 and CDG. While it is impossible to know why this happened, we can conclude that CloudFront policies are dynamic, in contrast with the static allocation of the EC2/S3 services. Furthermore, this highlights the need for persistent monitoring of traffic to highlight sudden changes.

Summary 4: *CloudFront cache in MXP has been replaced by FRA in 2013. Interestingly, however, despite the physical distance between end-users and the caches serving them has increased, the RTT and the number of ASes measured in the flows has decreased. Furthermore, we observe that the load distribution among CloudFront caches is dynamic, and can drastically change from a day to another.*



(a) 2012



(b) 2013

Figure 8: Distribution of response time ΔR for EC2 service in different periods.

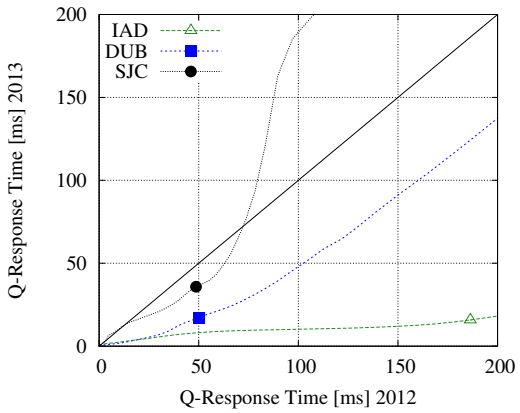
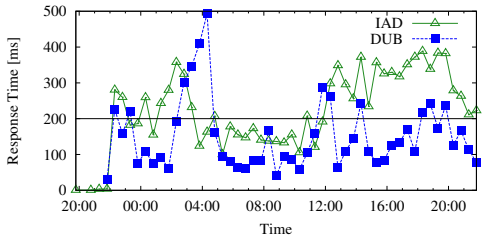


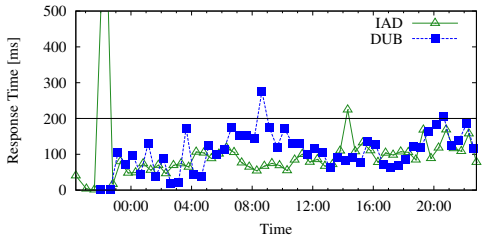
Figure 9: Q-Q plot of response time ΔR (2012 vs 2013) for EC2.

VI. Performance Monitoring

In this section we characterize the performance of AWS separately for each Availability Zone and, then, we focus on

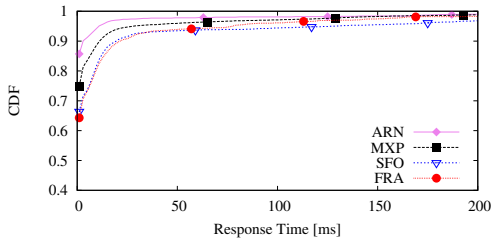


(a) 2012

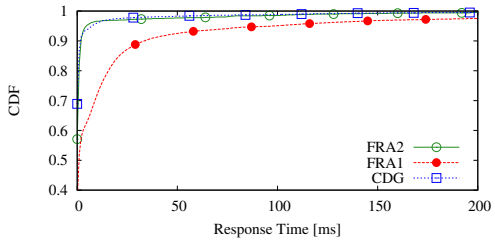


(b) 2013

Figure 10: Evolution over time of average ΔR for EC2 datacenters in different periods.



(a) 2012



(b) 2013

Figure 11: Distribution of response time ΔR for CloudFront service in different periods.

some specific web-services.

A. Availability Zone and Cache Response Time

Figs. 8a and 8b depict the distribution of the response time ΔR for EC2, in 2012 and 2013, respectively. We show the top popular datacenters. Focusing on 2012 dataset, IAD shows response time larger than 100ms in 30% of the cases, resulting the worst performing datacenter. In some cases, we observe badly engineered web-services that run on congested instances. For instance, web-service *proxy.eu.mydlink.com* served from DUB shows $\Delta R > 100$s during some periods. However, we observe such poor performance for many web-services running in IAD, suggesting a problem in the datacenter. In the same period, DUB and SJC represent a much better choice among Availability Zones.

2013 dataset presents a general improvement of performance in terms of response time: IAD shows a ΔR comparable to DUB. Instead, SJC has fallen behind. Fig. 9 which compares ΔR in 2012 and 2013 through the mean of a Q-Q plot, confirms such observation. If the curve falls in the area below the main diagonal, it reflects a better response time, worse otherwise. In this case, markers in the plot represent the 75th percentile of both distributions. The IAD 75th percentile in 2012 corresponded to 183ms, while it falls to 20ms in 2013. We conjecture that this improvement is due to the empowering of the datacenters as seen in Table II and Table III. The Q-Q plot shows also that web-services hosted on SJC datacenter have a slower reactiveness in 2013 than in 2012 only in the tail of the distribution.

We complement above results with Figs. 10a and 10b, which report the evolution over time of $E[\Delta R]$ for a period of one day for EC2 in IAD and DUB, for 2012 and 2013, respectively. Notice that the average response time is i) a highly variable measure (and possibly biased by the different web-services retrieved at different times), and ii) practically independent on the time of day.

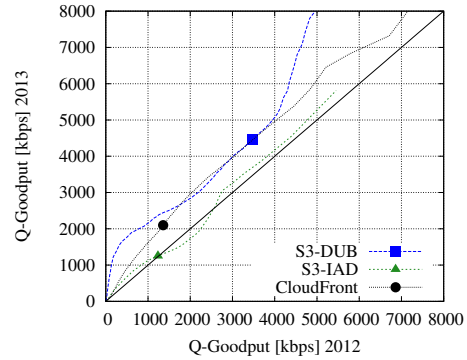
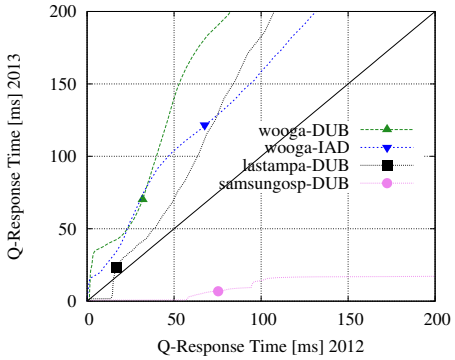


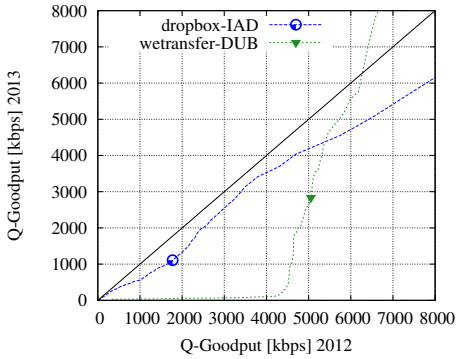
Figure 12: Q-Q plot of goodput G for the two most used S3 datacenters, IAD and DUB, and for CloudFront.

Moving to CloudFront, Figs. 11a and 11b show ΔR. In general performance is very good, being 83% of requests in 2012 satisfied in less than 20ms in FRA, the worst performing cache. MXP and ARN (Sweden) caches serve 80% of requests in less than 3ms; SFO and FRA serve only 65% of requests in less than 3ms, respectively. In 2013 we see a slight change in performance: cache FRA2, the most contacted by ISP end-users and the closest in terms of RTT, is the worst performing among the popular caches, with FRA1 and CDG offering the best reactiveness (90% of requests satisfied in less than 3ms).

Summary 5: *We observe a general reactiveness improvement in 2013 for Amazon datacenter hosting EC2. In particular, IAD has decreased its aggregated response time by a factor 2. CloudFront caches show in general a very good reactiveness, even if the most contacted cache in 2013 is the worst performing among the most popular ones.*

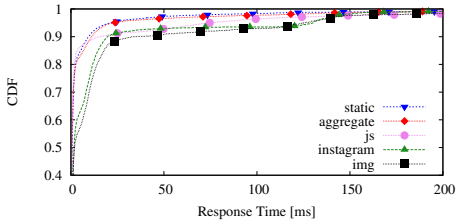


(a) Q-Q plot of response time ΔR for three main EC2 services.

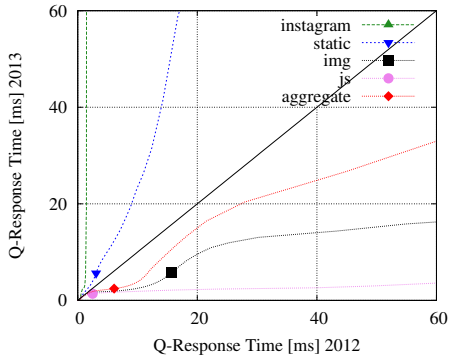


(b) Q-Q plot of goodput G for the two most important EC2 services in terms of volume.

Figure 13: Performance evaluation for several services. 2012 vs 2013 comparison.



(a) CDF (2012).



(b) Q-Q plot (2012 vs 2013).

Figure 14: Performance in terms of response time ΔR for different kinds of web-services which rely on CloudFront.

B. Availability Zone and Cache Goodput

We now focus on the evaluation of the goodput G for S3 and CloudFront services. Fig. 12 compares the distributions of goodput between 2012 and 2013, for S3 at IAD and DUB, and for CloudFront. Again we rely on the Q-Q plot with markers representing the 75th percentile. In this case a line above the main diagonal reflects an improvement. Some observations hold: first, the 75th percentile of goodput at DUB in 2013 is higher than 4Mb/s, while it is less 1.2Mb/s for S3 at IAD. This difference is very likely due to the large RTT that clients face to reach IAD (RTT >100ms); this affects the TCP congestion control, resulting in a limited goodput. Second, there is a improvement for S3 in DUB in 2013 with respect to 2012, while S3 in IAD exhibits stable performance. Third, despite the change of the cache location, i.e., from MXP to FRA, there is a considerable performance increase for CloudFront.

Summary 6: *The goodput for S3 in DUB has notably improved in 2013, as well as the goodput of downloads from CloudFront.*

C. Web-service Performance Monitoring

We now focus on the performance of specific web-services hosted by AWS.

Q-Q plot in Fig. 13a compares the distributions of response time ΔR of three popular web-services hosted by EC2 and deployed on IAD or DUB. The markers in the plot represent the 50th percentiles. Despite the general improvement of performance for EC2 between 2012 and 2013, especially in terms of reactiveness, surprisingly, users of popular web-services such as Wooga, Instagram and LaStampa experienced a significant decrease of performance, as their ΔR has almost doubled in one year. This demonstrates that a careful design and dimensioning of a web-service is a must for tenants, and poor performance is not always due to a lack of computational resources at the datacenter. SamsungOsp, instead, has notably improved its reactiveness.

Consider now the two web-services which generate the largest share of traffic volumes, i.e., Dropbox and WeTransfer. The Q-Q plot in Fig. 13b reports the comparison of their goodput. The markers in the plot represent the 75th percentiles. For both of them the goodput has decreased in 2013 with respect to 2012. This is in contrast with the observations of Sec. VI-B, where the goodput of traffic exchanged with DUB and IAD datacenters for S3 looks higher in 2013 than in 2012. The WeTransfer goodput decreases in 2013 might be related to the introduction of download rate limitations for free accounts.

At last, we analyze the performance of specific web-services that rely on CloudFront. Fig. 14a reports the distribution of ΔR for several kinds of web-service categories contacted by the end-users in 2012. *Static* refers to static content for web-pages (e.g., HTML files), *js* represents JavaScript files, *img* refers to binary data such as images, and *Instagram* is referred to contents related to the well-known photo-sharing web-service. For comparison, *Aggregate* reports the behavior of all web-services. As previously observed, CloudFront offers in general very good performance, being able to process 50% of requests in less than 2ms, independently on the kind of web-

service and web-object. However, ΔR is consistently smaller on average for static and JavaScript files, whereas images and Instagram contents show large response time. This may be due to the nature of the user-generated contents that are the most critical to manage for CDNs, because of the size of the catalogue, and of the small popularity of each single content [21].

We complement above results by analyzing the difference of performance between 2012 and 2013. The Q-Q plot in Fig. 14b (the markers report the 80th percentiles here) shows that the response time ΔR has improved in general, with specific contents such as images (*img*) and JavaScript (*js*) that are served faster in 2013 than in 2012. However, Instagram shows much worse performance (the 80th percentile is above 150ms). This may be caused by the increased popularity it acquired in Italy since its integration with Facebook which has started in mid 2012.

While it is hard to find the root cause of an evidence, the ability to monitor the performance of web-services running on Cloud and CDN infrastructures is a first step to understand implications and eventually trigger further investigations.

Summary 7: *Despite the aggregated reactiveness of EC2 has improved in 2013, the most popular services show a considerable performance decrease. A similar observation holds for the goodput of the two EC2 web-services which are responsible for generating the largest share of traffic. Finally, CloudFront suffers from lower reactiveness when serving user-generated contents. In particular, the performance of* Instagram *has dramatically worsened in 2013.*

VII. Related Work

Many measurement tools and platforms have been proposed in the last years, such as PerfSONAR [3] and RIPE Atlas [4]. RIPE Atlas only considers the case of active measurements, thus highly limiting its monitoring capabilities. While the measurement architecture we rely on, i.e., mPlane, might look similar to the one of PerfSONAR, the complete mPlane platform goes beyond, since it includes analysis of the gathered data to diagnose service and network problems. Furthermore, mPlane differs from those tools since it is specifically designed to adapt to the measurement task, making it extremely flexible. Moreover, the reasoner is a key component that allows structured, iterative, and automated analysis.

To the best of our knowledge, this is the first work which proposes a measurement platform based on passive observation of traffic for the monitoring of Clouds and CDNs. However, some frameworks which aim at monitoring and comparing the performance of different Cloud providers have been proposed in the past. Notable examples are [5], [22], [6]. All of them reside in the IaaS, and rely on active benchmarking to evaluate different performance metrics and rank Cloud providers accordingly to their results. Often IaaSes offer APIs themselves to provide a server-side monitoring interface on the internal state on the Cloud. E.g., Amazon CloudWatch or Azure Diagnostics let tenants monitor the virtual resource utilization and the performance of their deployments. Another approach is to collect reports made available directly from the tenants, as in SMICloud [23], or from users' [9]. Our platform is different, yet complementary to these systems, as, first, it operates from a perspective close to users', i.e., the monitoring platform is *outside* the monitored system. Second, it provides information that above solutions can not gauge: For instance, a tenant may be interested in understanding the impact of the datacenter choice on users' QoS.

To provide an idea about the benefits of our monitoring architecture, we focus the study in this paper on AWS, the most popular distributed Cloud infrastructure which incorporates many services in its eco-system, from Cloud computing to content delivery, in other words, the best candidate to provide an example of what our measurement architecture is capable of doing.

Since AWS is playing a key role in contemporary Internet, it has gained a large interest within the research community. In particular, many works investigated the possibility of exploiting AWS EC2 for research purposes [24], [25], [26]. Others instead focus on evaluating the performance of AWS computing and networking virtual resources [27], [28]. However, most of the previous works focus on the benchmarking of AWS IaaSes and infrastructure, and they all rely on "active" probing. To the best of our knowledge, the only work which aims at characterizing Amazon Web Services is [19], in which the authors base their study on DNS records and one passive trace. However, even if they consider other Cloud infrastructure providers in their study, they do not go beyond the characterization of Amazon EC2 and Windows Azure, and they provide no actual performance study.However, our results confirm many of their observations about Amazon EC2.

To the best of our knowledge, we are the first to provide an evaluation of actual AWS workload and performance by means of pure "passive" observation of traffic coming for a real operational network. Plus, our measurement architecture let us gauge the performance of web-services Cloud tenants run, and their impact on end-users. Finally, we are the first to analyze the evolution of a Cloud/CDN provider over a two-year long period.

VIII. Conclusions

Monitoring the traffic generated by Clouds and CDNs is hard as virtualization and content delegation deeply separates *web-services* from *server*, making it hard to understand what is actually delivered by whom.

To help untangle this web, in this paper we proposed a flexible measurement platform, whose design was specifically tailored to monitor traffic data generated by Clouds and CDNs.

We validated the effectiveness of our architecture by presenting a characterization of *Amazon Web Services* (AWS) traffic from passive measurements. By digging into the traffic generated by EC2, S3 and CloudFront over a period of two years, we show some interesting findings. For instance, there is a big workload unbalance among different datacenters hosting both EC2 and S3 products; in particular, the datacenter in Virginia is responsible for 85% of the total traffic sent to Italian end-users, despite the availability of a datacenter in Ireland.

We observed that in 98% of the cases tenants concentrate their content mostly on one datacenter, thus increasing the cost sustained by the network to carry data to faraway end-users. Considering AWS performance, we show that the datacenter in Ireland serves web-services more reactively and at higher transfer speeds than the datacenter in Virginia. However this latter was greatly upgraded between 2012 and 2013. We also found that CloudFront shows excellent performance, but i) Italian users are mainly served in 2013 by caches outside of Italy, and ii) it still presents issues that are typical of other CDN systems, especially when serving user-generated content.

We believe our proposal, the methodologies it implements, and the results we obtained from its validation can be useful for tenants, service-level agreement certificators, ISPs, as well as for the research community.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008.

[2] A. Finamore, V. Gehlen, M. Mellia, M. Munafò, and S. Nicolini, "The Need for an Intelligent Measurement Plane: the Example of Time-Variant CDN Policies," in *IEEE Networks*, October 2012, pp. –.

[3] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. apacz, D. Swany, S. Trocha, and J. Zurawski, "Perfsonar: A service oriented architecture for multi-domain network monitoring," in *Service-Oriented Computing - ICSOC 2005*, ser. Lecture Notes in Computer Science, B. Benatallah, F. Casati, and P. Traverso, Eds. Springer Berlin Heidelberg, 2005, vol. 3826, pp. 241–254. [Online]. Available: http://dx.doi.org/10.1007/11596141_19

[4] RIPE, "Ripe Atlas-Projects-RIS." [Online]. Available: http://www.ripe.net/db/index.html

[5] 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*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879143

[6] A. Turner, A. Fox, J. Payne, and H. Kim, "C-mart: Benchmarking the cloud," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1256–1266, June 2013.

[7] B. Trammell, P. Casas, D. Rossi, A. Bär, Z. Ben-Houidi, I. Leontiadis, T. Szemethy, and M. Mellia, "mPlane: an Intelligent Measurement Plane for the Internet," *IEEE Communications Magazine, Special Issue on Monitoring and Troubleshooting Multi-domain Networks using Measurement Federations*, Accepted for publication. [Online]. Available: https://www.ict-mplane.eu/sites/default/files//public/publications//756mplanerevised.pdf

[8] B. Trammell, M. Mellia, A. Finamore, S. Traverso, T. Szemethy, B. Szabo, D. Rossi, B. Donnet, F. Invernizzi, and D. Papadimitriou, "mPlane Architecture Specification," no. D1.3, Nov 2013.

[9] [Online]. Available: http://www.downrightnow.com

[10] [Online]. Available: http://www.downdetector.com

[11] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi, "Experiences of Internet Traffic Monitoring with Tstat," *IEEE Network*, vol. 25, no. 3, pp. 8–14, 2011.

[12] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP Geolocation Databases: Unreliable?" *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 53–56, Apr. 2011.

[13] I. Bermudez, M. Mellia, M. Munafò, R. Keralapura, and A. Nucci, "DNS to the Rescue: Discerning Content and Services in a Tangled Web," in *ACM IMC*, Boston, MA, November 2012.

[14] P. Mockapetris, *RFC 1035 Domain Names - Implementation and Specification*, Internet Engineering Task Force, November 1987. [Online]. Available: http://tools.ietf.org/html/rfc1035

[15] ARIN, "American registry for internet numbers." [Online]. Available: https://www.arin.net/

[16] L. Plissonneau, E. Biersack, and P. Juluri, "Analyzing the impact of youtube delivery policies on user experience," in *Proceedings of the 24th International Teletraffic Congress*, ser. ITC '12. International Teletraffic Congress, 2012, pp. 28:1–28:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2414276.2414310

[17] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of internet hosts," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 6, pp. 1219–1232, Dec 2006.

[18] Usenix, "Correlation between delay and distance." [Online]. Available: https://www.usenix.org/legacy/events/usenix02/full_papers/subramanian/subramanian_html/node21.html

[19] K. He, A. Fisher, L. Wang, A. Gember, A. Akella, and T. Ristenpart, "Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure," in *ACM IMC*, Barcelona, ES, November 2013.

[20] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, "Comparing DNS Resolvers in the Wild," in *ACM IMC*, Melbourne, AU, November 2010, pp. 15–21.

[21] B. Ager, F. Schneider, J. Kim, and A. Feldmann, "Revisiting cacheability in times of user generated content," in *IEEE INFOCOM*, San Diego, CA, March 2010, pp. 1–6.

[22] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, "What are you paying for? performance benchmarking for infrastructure-as-a-service offerings," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011, pp. 484–491.

[23] S. Garg, S. Versteeg, and R. Buyya, "Smicloud: A framework for comparing and ranking cloud services," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, Dec 2011, pp. 210–218.

[24] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientic Computing," *USENIX ;login: Magazine*, October 2008.

[25] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The Cost of Doing Science on the Cloud: The Montage Example," in *SC*, Austin, TX, November 2008, pp. 1 –12.

[26] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud," in *SAICSIT*, Wilderness, SA, 2008, pp. 94–103.

[27] S. L. Garfinkel, "An Evaluation of Amazons Grid Computing Services: EC2, S3, and SQS," Center for Research on Computation and Society, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, Tech. Rep., 2007.

[28] G. Wang and T. Ng, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," in *IEEE INFOCOM*, San Diego, CA, March 2010, pp. 1–9.