Cloud Storage Service Benchmarking: Methodologies and Experimentations

Enrico Bocchi
DET, Politecnico di Torino, Italy
enrico.bocchi@polito.it

Marco Mellia
DET, Politecnico di Torino, Italy
mellia@tlc.polito.it

Sofiane Sarni
LSIR, EPFL, Switzerland
sofiane.sarni@epfl.ch

***Abstract*—Data storage is one of today's fundamental services with companies, universities and research centers having the need of storing large amounts of data every day. Cloud storage services are emerging as strong alternative to local storage, allowing customers to save costs of buying and maintaining expensive hardware. Several solutions are available on the market, the most famous being Amazon S3. However it is rather difficult to access information about each service architecture, performance, and pricing. To shed light on storage services from the customer perspective, we propose a benchmarking methodology, apply it to four popular offers (Amazon S3, Amazon Glacier, Windows Azure Blob and Rackspace Cloud Files), and compare their performance. Each service is analysed as a black box and benchmarked through crafted workloads. We take the perspective of a customer located in Europe, looking for possible service providers and the optimal data center where to deploy its applications. At last, we complement the analysis by comparing the actual and forecast costs faced when using each service.**

According to collected results, all services show eventual weaknesses related to some workload, with no all-round eligible winner, e.g., some offers providing excellent or poor performance when exchanging large or small files. For all services, it is of paramount importance to accurately select the data center to where deploy the applications, with throughput that varies by factors from 2x to 10x. The methodology (and tools implementing it) here presented is instrumental for potential customers to identify the most suitable offer for their needs.

Keywords*—*Cloud storage, Web services, Amazon S3, Windows Azure, Performance measurement, Benchmarking, Comparison

I. Introduction

Nowadays, personal cloud storage services like Dropbox are customary among end-users [1]. Similarly, cloud storage services are gaining great popularity and evolving rapidly to address customers' requirements. An increasing number of potential users are being attracted by cloud storage solutions, which provide easy access to virtually unlimited storage and do not involve any set up cost. In this context, service providers own storage resources that are made available with a simple 'pay-as-you-go' billing model. Following the definition introduced in [2], storage providers offer the so-called *Infrastructure as a Service* as they partition and dynamically assign storage resources to customers according to their demand. The latter are mainly companies and mid-market businesses whose intention is to create and sell their own services without building and maintaining a dedicated hardware infrastructure.

Despite the increasing interest in storage and computing outsourcing, very little information is available as far as service architecture and design choices are concerned. To unveil such details, this paper presents methodologies and techniques that help storage services understanding. Firstly, service architectures are dissected in order to identify data centers locations and protocols involved. A simple geolocation methodology is used for the purpose, aimed at detecting actual deployment sites. Secondly, a benchmarking methodology is defined, which consists of a series of measurements that allow the assessment of each service performance under ad-hoc crafted workloads. A fair comparison is possible since all tests are executed taking the perspective of a customer located in Europe, who would like to identify i) the best provider; and ii) the best data center where to deploy his applications. An additional investigation on connection management policies is performed so as to emphasize the relevance of design choices for both end-users and the network load. Lastly, a comparison of pricing models is presented to highlight differences in service costs: predicted expenses for the considered workload are compared to actual charges invoiced with the aim of understanding whether it is possible to accurately forecast storage costs.

The strength of this work is the comprehensive comparison performed among four storage offers with benchmarks based on generic workloads, namely Amazon Simple Storage Service (S3)[1], Amazon Glacier[2], Windows Azure Blob[3], and Rackspace Cloud Files[4]. This is in contrast with previous works [3], [4] that include a single service and make use of specific project-driven workloads. The authors of [5] focus on measured bandwidth and bottlenecks related to the access network, but do not consider service architecture and pricing details. Finally, benchmarking tests here presented are based on active measurements, in contrast to [6], which relies on passive traffic analysis for Amazon cloud services characterization.

This paper provides a methodology to assess performance of cloud storage solutions, and to take an informed choice about which service and which data center better fulfils the application workload. We design and implement a set of tests to specifically evaluate performance of cloud storage solutions. We make available the benchmarking software to

This work is carried out in the context of the EU-IP projects *mPlane*: an Intelligent Measurement Plane for Future Network and Application Management (n-318627), and *CloudSpaces*: Open Service Platform for the Next Generation of Personal Clouds (n-317555), funded by the European Commission under its Seventh Framework Programme.

[1] *http://aws.amazon.com/s3/*
[2] *http://aws.amazon.com/glacier/*
[3] *http://azure.microsoft.com/en-us/services/storage/*
[4] *http://www.rackspace.com/cloud/files/*

TABLE I. PROVIDED AND ACTUAL DATA CENTER LOCATIONS AND NUMBER OF FRONT-END IP ADDRESSES

Amazon S3			**Windows Azure**		
Region Name (as provided)	Actual Location	IP addresses	Region Name (as provided)	Actual Location	IP addresses
US Standard	Washington D.C.	97	North US	Chicago	8
US West (N. California)	San José	18	South US	San Antonio	12
US West (Oregon)	Boardman	24	West US	Seattle	14
EU (Ireland)	Dublin	28	US East	Richmond	12
Asia Pacific	Singapore	6	Western Europe	Amsterdam	22
Asia Pacific	Sydney	2	North Europe	Dublin	20
Asia Pacific	Tokyo	5	East Asia	Hong Kong	4
South America	Sao Paulo	2	South Asia	Singapore	15
Amazon Glacier			**Rackspace**		
Region Name (as provided)	Actual Location	IP addresses	Region Name (as provided)	Actual Location	IP addresses
US East (N. Virginia)	Washington D.C.	3	Chicago	Chicago	1
US West (N. California)	San José	3	Dallas	Dallas	1
US West (Oregon)	Boardman	3	Northern Virginia	Washington D.C.	1
EU (Ireland)	Dublin	3	London	London	1
Asia Pacific	Sydney	3	Hong Kong	Hong Kong	1
Asia Pacific	Tokyo	3	Sydney	Sydney	1

the community upon request, so that everyone can run the tests with the desired workload using local connectivity.

We run the benchmarks from our Universities to provide useful insights on offers available on the market today, helping developers and customers in understanding their potential and eventual limits. Some of the collected results underline features and weaknesses of different solutions: no clear winner is eligible, as it changes depending on the used workload. For instance, Amazon S3 and Windows Azure Blob outperform Rackspace when coping with a lot of small files, but considering the download of few large files, Rackspace throughput tops to almost 300 Mb/s, versus 48 Mb/s and 40 Mb/s reached by Azure and S3, using the data center located in Dublin, Ireland.

II. METHODOLOGY

Cloud storage resources are accessible through interfaces directly exposed to the customer: a crude web-based management console, and an Application Programming Interface (API) based on REpresentational State Transfer (REST). Customers have thus to develop their storage management software according to the API. To design and implement the benchmarking methodology, we adopt the same approach, defining ad-hoc tools that generate synthetic workloads and obtain performance figures. API management is simplified taking advantage of a variety of Software Development Kits (SDKs) made officially available by storage providers. For our purposes, we used the Python SDKs shown in Tab. II. We focus on four available offers, even though the presented methodology is generic and can be applied to any other service. Storage solutions are selected according to their popularity on Google Trends[5] and are among those included in the Gartner "Magic Quadrant for Cloud Infrastructure as a Service"[6], August 2013.

[5] *http://www.google.com/trends/*

[6] *https://www.gartner.com/doc/2575715*

TABLE II. SERVICE SDK REVISION

Service	SDK version
Amazon S3	Boto, 2.19.0
Amazon Glacier	Boto, 2.19.0
Windows Azure Blob	Azure, 0.7.1
Rackspace Files	Pyrax, 1.6.2

A preliminary study has been performed to check which protocols are used, and if advanced features like compression, de-duplication or delta-updating [1] are eventually implemented to achieve storage and network optimization. Results show that all offers rely on proprietary protocols, carried over HTTPS/SSL, and no advanced capabilities are available. Some differences are present considering connection management and signalling traffic, as we briefly discuss in Sect. III.

A. Infrastructure discovery and geolocation

Each storage provider owns multiple facilities spread worldwide in order to guarantee reliability and data durability. Data center locations have also strong implications on performance and regulatory requirements. For instance, all storage providers suggest to select the closest end-point to the user to reduce network latency, and avoid TCP bottlenecks. For this reason, they make available a list of supported sites, but in most cases a coarse localization information is provided, e.g., using macro-regions granularity like West US, East Asia.

To both geolocate deployment sites accurately, and to disclose how these are reached from different locations worldwide, we identify a set of active experiments: first, hostnames of storage servers in each region are collected. Secondly, each server hostname is resolved querying more than 1,800 open DNS servers scattered around the globe (located in more than 100 countries, and involving 500 ISPs). This allows us to i) discover all front-end IP addresses; and ii) check if load-balancing techniques are implemented to route customers from different places to different IP addresses [7]. At last, geographical localization of server IPs is performed. As geolocation databases are known to be unreliable [8], we use a simple methodology based on multiple contributions:

• *Officially available information*: as provided by the service owner – see first column of Tab. I.

• *Airport codes*: by performing a reverse DNS lookup, Fully Qualified Domain Names (FQDNs) are retrieved and parsed to seek potential pieces of information revealing their location. As found in [9], server FQDN often embeds airport codes that are uniquely identified worldwide by using the International Air Transport Association (IATA) code.

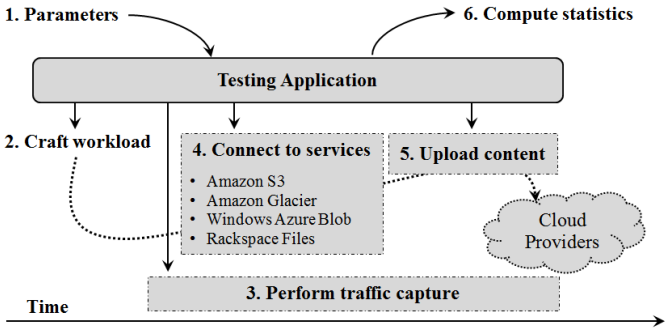


Fig. 1. Testing application diagram for content upload

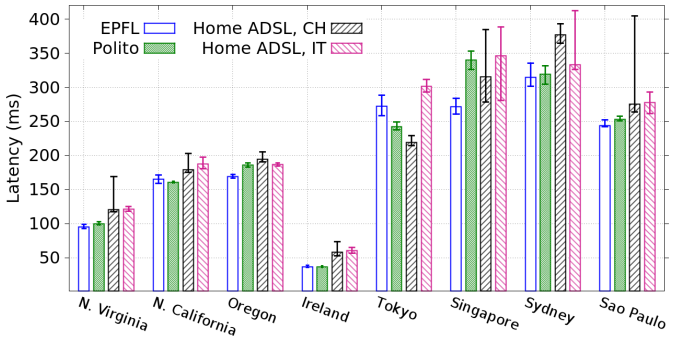


Fig. 2. RTT to Amazon S3 data centers

• *Closest server Round Trip Time*: Round Trip Time (RTT) measurements are performed from multiple PlanetLab nodes, whose deployment location is known. The position of the server returning the minimum RTT is then used as closest location to the storage server.

• *Traceroute to target hostname*: it allows the collection of domain names of intermediate routers. As before, router FQDNs can potentially contain information about their location (e.g., city names or IATA codes) that can be used to geolocate them and identify the closest to the targeted server.

As shown in [6], [9], these methodologies can provide an estimation of actual geographical location with a precision of about a hundred kilometres, which is suitable for our needs.

B. Testing application and workload

To benchmark selected services, a series of active measurements are performed. A testing application is designed for this purpose and plays the role of master controller during tests execution. Referring to Fig. 1, it receives multiple input parameters (step 1) for a proper: i) configuration of the environment (e.g., network interface to be used and remote credentials to access the service); and ii) definition of the workload. When testing content upload to the cloud, the following steps are performed: files are crafted according to workload specifications (step 2) and temporary stored on the local drive. Network traffic capture is started (step 3) to get a packet level trace that covers from session establishment up to the completion of all the uploads. This allows us to record login and pre-upload negotiation steps as well as actual content transfer. Authentication with the storage service is accomplished (step 4) and, finally, content is uploaded to the cloud (step 5). For each transferred content, the application generates and logs some metadata, which are inspected at a later stage (step 6) together with traffic captures to compute desired performance metrics. After a file has been uploaded, the application waits for two minutes, and then downloads the same file following similar steps.

The desired workload is crafted varying i) file sizes; and ii) the number of files to be transferred to the cloud. A great variety of prospected usages is thus considered, leading to seven different benchmarking sets composed as follows:

• *Single files*: four files are defined with sizes equal to 1 MB, 10 MB, 100 MB and 1 GB. Each of them is transferred as free-standing workload.

• *Bundles of files*: each bundle is composed of multiple small files that are transferred in a single transaction. Three bundles are defined: i) 10 files of 100 kB each; ii) 100 files of 10 kB each; and iii) 1,000 files of 4 kB each.

In all cases, files used as payload are filled by randomly generated bytes. Each experiment is repeated twenty times per service, with a three minutes' idle time between consecutive tests. A full cycle takes approximately ten hours to complete, with a total amount of exchanged data equal to 45 GB.

III. Results

In this section we present results collected by running tests during December 2013 and January 2014. They thus reflect a snapshot of what the systems were at that time. Experiments were executed from EPFL in Lausanne and Politecnico di Torino (Polito). The servers used for testing are connected to a 1 Gb/s Ethernet LAN, and to the public Internet with 10 Gb/s campus access links. Local connectivity cannot be considered a bottleneck in any of performed tests, as confirmed by results. In general, outcomes from EPFL and Polito are very similar. As such, we mainly report measurements collected from EPFL.

A. Data center locations

Tab. I shows service region names together with actual deployment sites and the amount of front-end server IP addresses in each region. According to results, Amazon S3 and Windows Azure rely on eight different data centers each, while Amazon Glacier and Rackspace have six data centers available. Despite using thousands of DNS servers scattered worldwide, we did not notice differences in resolving server IP addresses, i.e., each server name is always resolved to the same IP addresses independently on the client location. Looking at the number of IP addresses matching each data center, we observe that for Amazon S3 and Windows Azure different regions present a different amount of IPs, with Washington D.C. and Amsterdam being the highest respectively. Glacier always uses only three front-end IPs in all locations, while Rackspace exposes resources from a single IP address, which may pose reliability issues in case of, e.g., routing problems.

Data center locations have severe implications on network latency. Considering Amazon S3 regions for example, Fig. 2 details the observed RTT[7] from four different access scenarios:

[7]RTT has been measured both using ICMP (ping) and TCP (nmap) handshake messages.

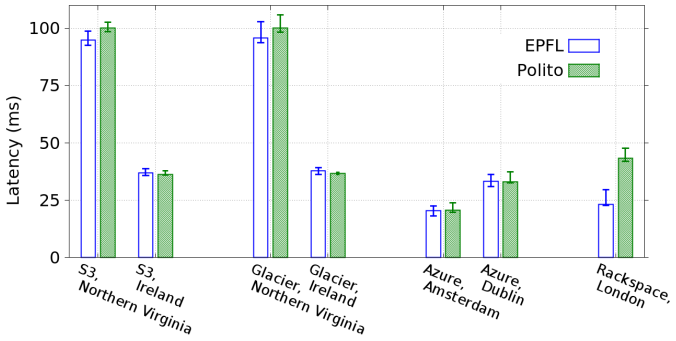


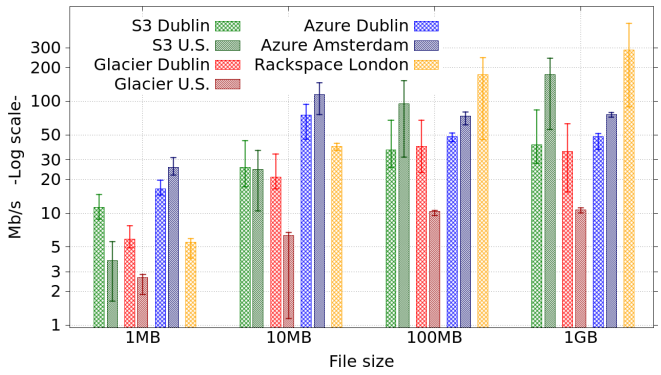
Fig. 3. RTT of data centers selected for benchmarks

two are Universities campuses, and two are home ADSL subscriptions in Italy and Switzerland. Latencies are clearly dominated by the propagation delay, with home connectivity suffering of a slightly higher delay due to slow access networks. Minimum and maximum RTT observed over a hundred experiments are reported using error bars. As it can be seen, measurements are consistent suggesting stable and uncongested paths. The closer the data center, the smaller the variation. Results from other services are similar (see also Fig. 3) and not reported for the sake of brevity.

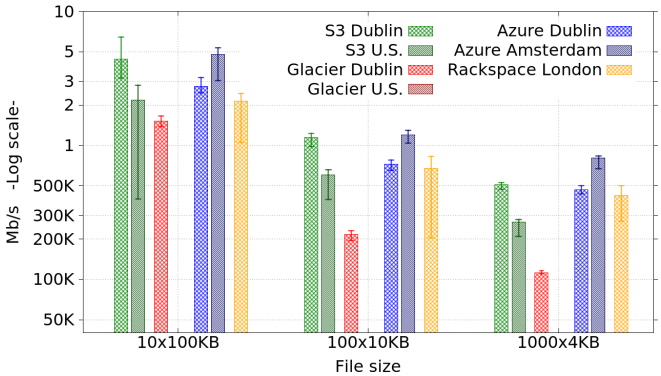
Storage providers strongly encourage users to select the closest end-point to their location in order to reduce data latency and to offload the public Internet. Therefore, we selected the geographically closest data centers with respect to our Universities, picking Dublin for Amazon services and Windows Azure, and the London site for Rackspace. For comparison purposes, we also benchmarked other data centers: Washington D.C. for Amazon, and Amsterdam for Windows Azure. Fig. 3 summarizes and compares measured RTTs for selected facilities. As before, measurements are very stable and mostly driven by propagation delay. Some differences are visible, with EPFL having shorter RTTs than Polito, owing to different network paths.

B. Transfer throughput

We now focus on the performance offered by services and data centers. Let us start by considering the throughput observed when downloading a single file. Fig. 4(a) reports results considering file sizes of 1 MB, 10 MB, 100 MB and 1 GB. Notice the log scale on y axis. Several considerations hold. First, download rate improves for big files. This is a well-known effect caused by TCP slow-start and congestion control algorithms that require several RTTs to allow the growth of the congestion window. Second, Amazon Glacier is by far the slowest service: Dublin data center seems to limit the download rate to about 40 Mb/s, while the Washington site can barely reach 10 Mb/s. Comparing Amazon S3 and Glacier it is evident that, despite being both hosted in the same data center, the download rate of Glacier is artificially enforced. Indeed, S3 tops to about 175 Mb/s in the Washington site, approximately 18 times faster than Glacier. Third, there is no clear winner: for small files Windows Azure guarantees the highest throughput, but for large ones Rackspace reaches almost 300 Mb/s. Comparing the two Amazon data centers, Dublin outperforms Washington with short files, as the much higher RTT of the latter severely impairs the TCP slow-start



(a) Single-file workload



(b) Multi-file workload

Fig. 4. Download throughput for selected data centers and services

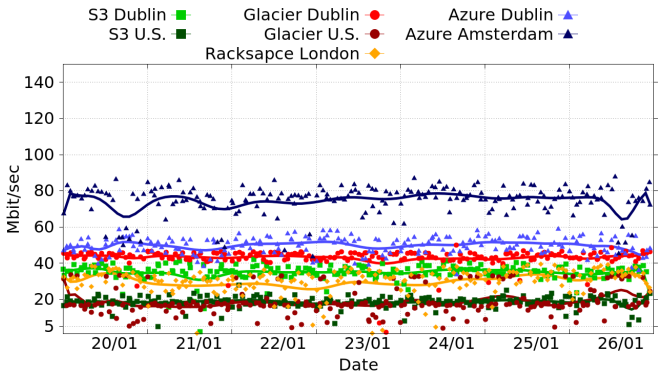
algorithm. On the other hand, Washington shows to be much faster for large files (suggesting a better connectivity and path). Moving to Windows Azure, the Amsterdam data center is preferable to the Dublin one, possibly due to the lower RTT (see Fig. 3). The same effect is visible when considering the upload direction (not reported here due to lack of space).

Let us now consider a different workload according to which a number of files has to be downloaded from the cloud. Fig. 4(b) reports outcomes considering different combinations of number of files and sizes. First, notice that Amazon Glacier in Washington did not allow to run this experiment.[8] Second, observe how the number of objects has a huge impact on performance. The bottleneck here results to be the signalling for cloud management rather than the actual capacity of the path. Indeed, the introduced overhead grows with the number of objects, and can affect the throughput, lowering it to less than 700 Kb/s when downloading thousands of small objects.

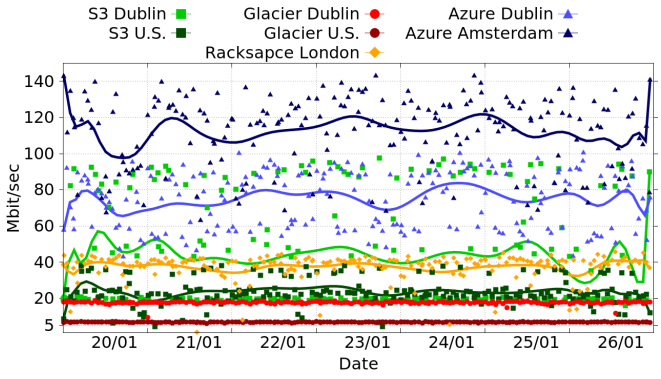
C. Persistent monitoring

We present now a measurement campaign collected by repeating experiments over time in order to understand if cloud services present typical periodicity due to, e.g., traffic peaks during some hours of the day, which can potentially cause a performance drop. To this purpose, we consider a workload in which a 10 MB file is uploaded and downloaded repeatedly. A total of 3,500 single transactions are performed in the week

[8]Glacier is a service mainly targeting the archival of large files. It is however unclear why it does not support the "restore" of a batch of files.



(a) Upload direction



(b) Download direction

Fig. 5. Time dependency assessment of throughput for different services and target data centers

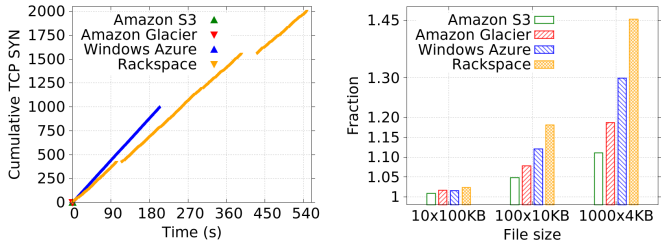
from Monday 20th to Sunday 26th of January 2014. Fig. 5(a) and Fig. 5(b) show measurements for upload and download directions respectively. Dots report single measurements while curves are the smoothed averages.

Considering the upload, all services exhibit an almost stable performance with no evident correlation with the time of the day. The download is typically faster than upload except for Glacier where it is evident the artificial limitation of the available bandwidth, already observed before. Performance changes widely over time with still no correlation with the period of the day. Windows Azure in Amsterdam performs the best, peaking at 140 Mb/s. Amazon S3 in Dublin presents a huge variation, jumping between 20 Mb/s and 100 Mb/s almost with a random fashion. Investigating further, we identify two groups of servers belonging respectively to the subnet 178.236.0.0/21 and 54.239.34.0/22. Both are reached using the same path, but the first group offers throughput higher than 80 Mb/s while the second is clearly limited to 20 Mb/s. The application has no control on which server will be returned.

In summary, results show that there is a great variability in performance, considering different i) services; ii) sites; and iii) workloads. In some cases, variations are not controllable by the application, but due to cloud provider policies. This underlines the need to run proper benchmarks before choosing an offer.

D. Connection management and overhead

We now report some additional insights about different implementation choices made by cloud providers. All selected



(a) TCP-SYN for 1,000 files (b) Overhead

Fig. 6. Connection management and overhead for multiple files upload

services make use of a single remote server per transaction that is in charge of both user's authentication, and content transfers. Similarly, all services implement multiple transfers in sequence, i.e., the next transfer operation is started only after the previous one has been completed. However, a variety of connection management policies can be identified by analysing network traffic. We consider a scenario where multiple files are being transferred and we monitor the number of HTTPS connections established and the total cost for the network, including overheads added by SSL and application layers.

Considering the upload direction, Amazon Glacier is the most cleverly implemented service: it makes use of a single connection for both user login and content transfer. Amazon S3 shows a similar behaviour opening two connections, one for control and one for actual data exchange, this independently on the number of files to be transferred. Windows Azure opens instead a single control connection, but it uses one TCP (and SSL) connection for each transferred content. Finally, Rackspace opens two separate connections per file in addition to three control connections established at the beginning of the transaction. Fig. 6(a) clearly highlights this by reporting the amount of `TCP-SYN` segments observed when 1,000 files have to be uploaded. It also shows that both Windows Azure and Rackspace open connections sequentially.

While it may not result critical for the application, this has noticeable implications on the generated overhead that the network has to carry. To quantify it, Fig.6(b) reports the ratio between the amount of payload carried by TCP connections and the workload size. It clearly shows that Rackspace policy generates almost 50 % additional overhead when uploading lots of small files. Windows Azure pays the SSL overhead cost as well, while Amazon S3 and Glacier are far less affected by bandwidth wastage, with overhead down to 11.08 % and 18.72 % respectively, thanks to using only one TCP/SSL connection. This extra-cost can be not negligible especially considering applications developed for mobile devices.

IV. Pricing Model

All selected storage offers are based on 'pay-as-you-go' subscriptions with no fixed costs. Three main types of variable expenses are charged:

• *Raw storage:* Amazon S3 and Windows Azure start charging 0.085 $ per GB per month. Rackspace is slightly more expensive (0.10 $/GB/month), while Amazon Glacier is by far the cheapest (0.01 $/GB/month). Prices decrease along with the amount of storage needed for all services but Glacier, which has a constant price independent from the volume used.

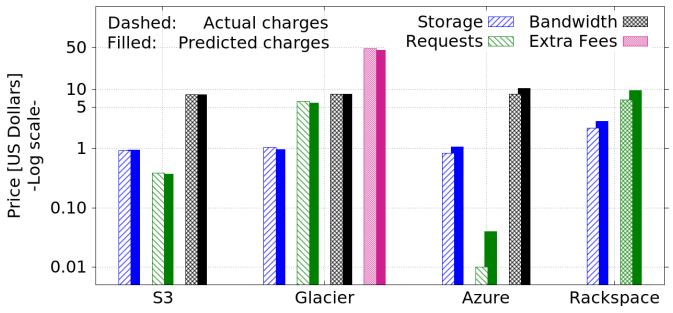


Fig. 7. Actual vs. Predicted charges

• *Requests:* all services but Rackspace charge proportionally to the number of operations. Amazon S3 charges 0.005 $ per 1,000 PUT, COPY, LIST requests and 0.004 $ every 10,000 GET requests. Windows Azure is cheaper, charging 0.005 $ per 100,000 requests. Amazon Glacier is the most expensive given its long-term archival nature (0.05 $ per 1,000 requests).

• *Outgoing bandwidth:* upload to the cloud is free, but download is charged by volume. All services adopt similar prices, starting at 0.12 $ per GB and decreasing with the amount of capacity consumed.

Amazon Glacier involves additional fees in case contents are deleted prior than ninety days or accessed frequently. Users are allowed to retrieve up to 5 % of stored data for free. If this threshold is exceeded, the so-called *peak retrieval rate* is computed as difference between the amount of data requested and the free allowance in a four hours' time. An extra cost of 0.01 $ per GB is then charged for the whole month duration (i.e., *peak retrieval rate* $\times$ 0.01 $ $\times$ 24 $\times$ 30, assuming a 30-day month). Amazon Glacier is indeed a service prospected to long-time archival, where downloads are unusual.

Usage costs depend on the workload the application produces and predicting them is hard. We instead compare costs charged by service providers with respect to expected costs, computed knowing the exact workload we generated. This allows us to understand whether it is possible to forecast storage charges accurately, and eventually identify unexpected peaks. Fig. 7 shows the results splitting costs among the three categories previously detailed and adding extra fees invoiced by Glacier.

In general, predicted and actual charges are similar with differences below 5 % for Amazon services. Windows Azure and Rackspace tend to charge less then expected in both storage and bandwidth (about 25 % less). For our workload, bandwidth costs dominate, being more than twenty times the actual storage cost. In our benchmarks, indeed, files are removed from the cloud once the test has been completed in order not to waste remote storage and reduce benchmarking expenses. Considering request costs, Windows Azure cheaper policy is evident with respect to Amazon. Glacier results to be very costly due to the expensiveness of download operations that, in our case, implied a huge extra fee. Indeed, we generated a workload that is highly discouraged by the storage model Glacier offers, i.e., retrieval operations are very expensive.

V. Conclusions

In this paper we presented specific methodologies to shed light on cloud service architectures, assess end-user performance and verify storage charges. Such methodologies are then applied to benchmark four offers from the point of view of an European customer. No clear winner is evident.

All services have multiple data centers available in the US, in the Far East and in Europe. The latter location presents advantages when transferring lots of small files, since network latency plays a relevant role. For large files, however, throughput on US data centers can be higher than the European ones. Surprisingly, we noticed artificial throughput limitations imposed by S3. These are not predictable, being neither correlated with the selected data center, nor with time of the day. Considering protocols overhead, a relevant factor when designing mobile applications, Amazon services implement a smart connection management that limits network load. Windows Azure and Rackspace can have up to 50 % of additional overhead instead. As far as storage expenses are concerned, all services show to have similar pricing models. It is possible to forecast expenses with good accuracy, provided to know the exact workload faced by the storage solution.

In summary, our methodologies and tools prove to be useful to benchmark cloud storage offers, and make informed choices when looking for the most suitable storage provider for customers' needs.

References

[1] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 205–212. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504762

[2] 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. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[3] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: A viable solution?" in *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing*, ser. DADC '08. New York, NY, USA: ACM, 2008, pp. 55–64. [Online]. Available: http://doi.acm.org/10.1145/1383519.1383526

[4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, Nov 2008, pp. 1–12.

[5] A. Bergen, Y. Coady, and R. McGeer, "Client bandwidth: The forgotten metric of online storage providers," in *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*, Aug 2011, pp. 543–548.

[6] I. Bermudez, S. Traverso, M. Mellia, and M. Munafo, "Exploring the cloud from passive measurements: The amazon aws case," in *INFOCOM, 2013 Proceedings IEEE*, April 2013, pp. 230–234.

[7] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan, "Mapping the expansion of google's serving infrastructure," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 313–326. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504754

[8] 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. [Online]. Available: http://doi.acm.org/10.1145/1971162.1971171

[9] B. Eriksson and M. Crovella, "Understanding geolocation accuracy using network geometry," in *INFOCOM, 2013 Proceedings IEEE*, April 2013, pp. 75–79.