On the Performance of Cloud Storage Applications with Global Measurement

Guangyuan Wu[1] Fangming Liu*[1] Haowen Tang[1] Keke Huang[1]
Qixia Zhang[1] Zhenhua Li[2] Ben Y. Zhao[3] Hai Jin[1]
[1]Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology
[2]School of Software, TNLIST, and KLISS MoE, Tsinghua University
[3]Department of Computer Science, University of California Santa Barbara

***Abstract*—In recent years, Dropbox, Google, and Microsoft have been competing in the market of consumer cloud storage (CCS) services. While once the key comparative metric, storage capacity per user has outgrown the needs of most users. Today, third-party applications based on CCS's RESTful Web APIs are becoming a primary way for users to utilize their expanded storage resources. Unfortunately, there is very little visibility into the performance of these Web APIs, even though they are primary determinants of the end user experience on these storage applications. In this paper, we report results from a comprehensive measurement study of the Web APIs of five popular CCS providers. Our results reveal significant differences and limitations in API performance, which result in performance bottlenecks visible to the user through the storage application. We analyze the underlying system designs of the five providers' Web APIs, and present the performance implications of their different design choices. Our research provides practical guidance for service providers to optimize their API performance, for developers to improve the experience of third-party applications, and for users to pick appropriate services that best match their requirements.**

I. Introduction

Consumer cloud storage (CCS) services like Dropbox provide convenient ways for users to access and share their files stored in the cloud. With automatic data synchronization, these services provide simple backup and synchronization between devices. Recent years have seen the rapid proliferation of these services, with competing offers from industry giants such as Microsoft and Google joining the fray (*i.e.*, OneDrive and Google Drive). Growing competition in this sector has led to lower costs and a commoditization of the service, with both Dropbox and Google Drive offering 1 TB of storage for $9.99 per month. The competition is even more heated in China, where providers eager for new users offer nearly unlimited capacity (up to tens of terabytes per user) for free.

It is becoming clear, however, that larger storage capacity is no longer a strong competitive advantage. Today's users are finding it challenging or difficult to fully utilize the seemingly limitless capacity offered by Dropbox and Google Drive, let alone the 36 TB offered by 360Cloud. A quick look at user forums turns up comments by users unclear on how to use the large storage capacities [1].

Instead, a variety of popular third-party applications based on CCS are becoming pervasive. For example, URL Droplet helps users save interesting online content directly to Dropbox. For people who take voice memos, DropVox uploads them to users' Dropbox automatically. Password keepers like 1Password store encrypted data on Dropbox, and powerful Web apps like WeVideo, Audio Cutter, and PicMonkey store edited videos, audios, and photos directly to Google Drive and other CCS services. As these applications continue to grow, CCS's RESTful Web APIs, which provide storage access for these third-party applications, are now becoming one of the major ways for users to access their expanded storage.

The performance (particularly average throughput, delay, and failure rate) of CCS Web APIs directly impact the user experience on any third-party applications they serve. While prior work has studied the performance of CCS native clients [1], [2], [3], [4], [5], [6], there is very little visibility into the performance of CCS Web APIs. In this paper, we address this through an empirical study of Web APIs of five popular CCS providers, including Dropbox, OneDrive, and Google Drive in the U.S., and BaiduPCS and DBank in China. We separate APIs into data APIs for file uploading and downloading, and control APIs for file and folder operations. Using a recent trace of CCS users' file operations [6], we develop methodology for a long-term global measurement, and deploy a benchmark system on 13 PlanetLab vantage points covering 9 countries across 5 continents (§ III). We analyze different system designs of the five providers' Web APIs, and discuss the key factors that impact the API performance. We list our key findings below:

- **Underlying system design** (§ IV). While the five CCSs provide similar functionality through the Web APIs, the underlying system designs (*i.e.*, API protocol, server location, and load-balancing strategy) are very different. We show that only Dropbox uses a centralized system design, and Google Drive relies on its content delivery ecosystem to carry much of its user traffic. All these factors directly

*The Corresponding Author is Fangming Liu (fmliu@hust.edu.cn). The research was supported in part by the National Natural Science Foundation of China (NSFC) under grant 61520106005, National 973 Basic Research Program under grant 2014CB347800, and National 863 High-Tech Research and Development Program under grant 2015AA01A201.

[1]https://www.dropboxforum.com/hc/communities/public/questions/201370529-WHY-1TbB-

impact API performance.

- **Data API: Network bottleneck**. Network plays a significant role in data API performance. Since Dropbox hosts its data APIs only in Amazon's North Virginia data center, the performance for transfer of small files of MBs is very unstable across different locations because of diverse RTTs (§ V-B). And network congestion significantly depreciates performance of the data APIs of BaiduPCS and DBank (§ V-E). In contrast, OneDrive and Google Drive use different geo-distributed system designs to mitigate the limitation of network in different degrees (§ V-C and § V-D).
- **Data API: Protocol overhead** (§ V-C). Although geo-distributed system design improves the performance of data APIs, the design choice brings extra complexity in API protocol. In our measurement, the complex protocol may introduce non-negligible overhead in certain cases.
- **Control API: Server processing time** (§ VI-A). Server processing time greatly impacts the delay of control APIs. Most control API requests only need a few of RTTs for message exchange. Thus server processing time makes up a large part of end to end application delay. We quantify server processing time of each control API of the CCSs (except Google Drive), which can help third-party application developers to design appropriate business logic for interactive applications.

Our results reveal significant differences in performance of different CCSs' Web APIs across locations and time, and analyze the substantial underlying differences among providers. Our work provides practical guidance for users to pick appropriate services that best match their requirements, while suggesting possible directions for service providers to optimize system design, and for third-party developers to improve application performance.

II. CCS API Overview

Since third-party applications are becoming pervasive in people's everyday life, CCS RESTful Web API has already become one of the major ways for users to access their expanded storage. In this section, we first summarize the capabilities provided by five popular CCSs' Web APIs. Then we compare CCS Web APIs with native clients from several aspects to shed light on the essential differences between them.

A. Capabilities Provided by CCS Web APIs

Table I provides a summary of the RESTful Web APIs provided by five popular CCS providers (including Dropbox, OneDrive, Google Drive, BaiduPCS, and DBank). We can see that all providers offer simple file upload/download (data APIs) and basic file/folder operations (control APIs, *e.g.*, `create`, `list`, `move`, `copy`, and `delete`).

However, several points need to be noticed. First, in terms of file transfer capability, Dropbox and BaiduPCS support uploading large files in multiple trunks, which provides the same resumable upload feature as Google Drive does, and BaiduPCS further enables applications to avoid actual data transmission if the content already exists in the cloud. Second,

TABLE I
Capabilities of Five CCSs' Web APIs

Web APIs	Service Providers
Basic Functions (upload, download, create folder, list, move, copy, delete)	Supported by all five providers
Search	Not supported by DBank
Revisions	Not supported by OneDrive and BaiduPCS
Share	Not supported by BaiduPCS
Check update	Not supported by OneDrive and DBank
Notification	Not supported by OneDrive, BaiduPCS, and DBank

in terms of file/folder operations, OneDrive and Google Drive do not support copying folders. Presumably because that OneDrive and Google Drive use a storage model based on file IDs rather than a traditional file hierarchy used by other three CCSs. Considering file sharing function, only BaiduPCS does not provide the `share` API. Last but not least, both Dropbox and Google Drive provide advanced capabilities like `check update` and `notification` to enable easy implementation of synchronization function for third-party applications.

In a word, for developers intend to build applications on top of CCS APIs, it is necessary to take into consideration the choice of the providers due to their diverse API capabilities.

B. Differences between Web APIs and Native Clients

Generally, the way for users to access a CCS's official service is its native client, which provides user-friendly graphical user interface with commonly used capabilities like automatic data synchronization. Previous works studied the underlying protocol used by CCS native clients [1], [4], [2], [3], [6]. However, three major differences exist between CCS Web APIs and native clients: 1) Different purposes. CCS providers develop their own proprietary native clients and design advanced client protocols to facilitate efficient file synchronization, while their Web APIs are offered to developers to build up third-party applications with diverse functionalities to enrich their services. 2) Different capabilities. Native client protocols are specialized for file synchronization with complex and stateful logics, while Web APIs provide general and simple data access capabilities in a RESTful (*i.e.*, stateless) fashion as we discussed above. 3) Different system designs. Web APIs not only have simpler application layer protocols, but also are served by a different set of service domains compared to their native clients [1].

These huge differences imply that the performance of CCS Web APIs is deemed to be very different from that of native clients. This naturally raises several questions. *How do CCS providers design their RESTful Web APIs? How well can different CCS APIs perform? What are the key factors that affect the performance of CCS APIs?* The remainder of this paper answers these questions.

III. Methodology

In this section, we describe how we design and conduct global measurement to analyze and compare the performance of different CCSs' Web APIs. We first demonstrate how we implement our system to benchmark CCS Web APIs. Then we show the measurement setup, including how we deliberately select popular CCS providers, the most common and useful APIs, crucial performance metrics, and globally distributed vantage points. In addition, we illustrate how we deploy the measurement on a global scale with RTT measurement and how we find the locations of the CCSs' API servers.

A. Benchmark Tool

Most CCS providers offer their APIs via HTTP-based RESTful APIs. To make an API call, a third-party application should generate an HTTP request message conforming to the API's protocol and send it to a service domain (a fixed domain provided in the official API documents or a temporary one returned by a previous response). Then the application receives an HTTP response message which contains file content of a download API call or tells the result of an upload or control API call. Although most APIs only require one request/response exchange in an API call, multiple exchanges may be necessary for several APIs.

Instead of using existing SDKs offered by the CCS providers, we develop a measurement program from scratch using C++ for a fair comparison among CCSs and full control over the measurement. The program consists of three layers. The bottom layer is an HTTP(s) client built with Socket APIs and OpenSSL APIs. This layer records domain names and IPs used in each API call for an in-depth understanding of the underlying system designs. The middle layer contains five modules to implement the API protocols of the five CCSs. Specifically, the modules generate HTTP request messages, send them via the HTTP(s) client, and receive and parse the HTTP response messages. The top layer calls a specific API according to the command line arguments, and records the delay and result into log files.

B. Measurement Setup

Selection of CCS providers. We deliberately select five representative CCS providers worldwide: Dropbox, OneDrive, Google Drive, BaiduPCS, and DBank. The former three are the most popular CCS providers who have the largest user base, the latter two are popular products from IT giants in China. Most of these CCSs provide terabytes of storage at a fairly low price or even for free (*e.g.*, BaiduPCS offers 2 TB storage for free accounts), which is extremely attractive to both users and third-party developers.

Selection of API sets. Despite diverse API capabilities offered by different CCS providers, there exist two common core sets of APIs that lay a foundation for implementing a full-fledged CCS application, *i.e.*, basic file transmission (data APIs), including `upload`, `download`, and basic file/folder operations (control APIs), including `create folder`, `list`, `move`, `copy`, `delete` and `share`. These APIs are offered as the basic functionalities by most CCSs today as evidenced in Table I. Although a few CCS providers offer specialized APIs such as `notification`, we skip evaluating these special offerings and focus on the more general and basic APIs.

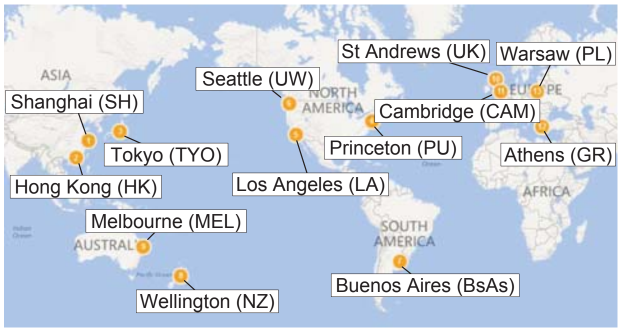


Fig. 1. 13 vantage points covering 9 countries across 5 continents.

Selection of metrics. Three metrics are devised to measure the API performance: **average throughput** (*i.e.*, the average data rate for transferring a file, which is a key indicator of data API performance), **delay** (*i.e.*, the time it takes for a specific API to complete, which is the most intuitive user experience of all APIs), and **failure rate** (*i.e.*, the ratio of failed API requests since not every API call will be successful). These metrics reflect the performance of CCS APIs, which is essential for us to reveal the performance implication of the CCSs' different system designs.

Selection of vantage points. PlanetLab is a global research network that has been used as the experimental platform for tremendous networking and measurement researches. Since PlanetLab nodes are shared virtual machines contributed by different universities, there may exist performance issues such as other applications or background traffic that will compete for the bandwidth (such issues also exist in normal usage of CCSs). To minimize such interference, we use speedtest-cli [2], a bandwidth test tool based on python, to test the bandwidth limitations on different PlanetLab nodes. We select 13 geo-distributed nodes covering 9 countries across 5 continents (see Fig. 1), which steadily provide over 40 Mbps upload and 100 Mbps download bandwidth, so as to obtain the actual end-to-end networking performance when accessing CCS APIs from different locations.

Measurement deployment. For a comprehensive understanding of CCS API performance across locations and over time, we deploy our measurement system on the aforementioned 13 PlanetLab nodes, and periodically (every half an hour) measure each control and data API for all the five CCSs during 20 consecutive days in April, 2015. Specifically, for control API, we perform file/folder creation, movement, copy, delete, list, and share in sequence; for data API, we take turns to upload and download files with given sizes (0/0.5/1/2/4/8/50 MB) [3] to and from each CCS back to back to ensure fair networking conditions. Note that we use randomly generated files in our measurement to avoid possible

[2]https://github.com/sivel/speedtest-cli/

[3]We choose these sizes because 1) most CCS users transfer files of MBs [1], and 2) some CCSs like Dropbox only support 150 MB data in one upload API call.

TABLE II
UPLOAD API PROTOCOLS

CCS	Service domain	SSL	Request/Response
Dropbox	api-content.dropbox.com	✓	File/Metadata
OneDrive	apis.live.net	✓	File/Metadata
Google Drive	www.googleapis.com	✓	File/Metadata
BaiduPCS	c.pcs.baidu.com	✓	File/Metadata
DBank	api.dbank.com	✗	Request/Server IP
	temporary IP	✗	File/Metadata
	api.dbank.com	✗	File path/Result

TABLE III
DOWNLOAD API PROTOCOLS

CCS	Service Domain	SSL	Request/Response
Dropbox	api-content.dropbox.com	✓	File path/File
OneDrive	apis.live.net	✓	File ID/File URL
	temporary domain	✓	GET/File
Google Drive	www.googleapis.com	✓	File ID/File URL
	temporary domain	✓	GET/File
BaiduPCS	d.pcs.baidu.com	✓	File path/File URL
	temporary domain	✗	GET/File
DBank	api.dbank.com	✗	File path/File URL
	temporary domain	✗	GET/File

TABLE IV
CONTROL API PROTOCOLS

CCS	Service domain	SSL	Capabilities
Dropbox	api.dropbox.com	✓	Create folder;
OneDrive	apis.live.net	✓	Move file/folder;
Google Drive	www.googleapis.com	✓	Copy file/folder;
BaiduPCS	pcs.baidu.com	✓	Delete file/folder;
DBank	api.dbank.com	✗	List; Share

interference from existing files in the systems. In addition, we use different accounts in all vantage points in order to simulate the real workload generated by unrelated individuals.

In order to facilitate the analysis, we interleave the API tests with RTT measurements to the domain names in the service URLs. Specifically, for each domain name, we repeatedly conduct a TCP SYN probing for 10 times, *i.e.*, sending a TCP SYN segment to the HTTP(s) port on the resolved server IP. The server will then return a TCP ACK segment, which tells the RTT between the vantage point and the server [4]. From the result of RTT measurement, we can obtain information about the network condition between each vantage point and server.

To reveal the server locations of the five CCSs' Web APIs, all IP addresses collected in our measurement are queried against several online databases. Furthermore, we also use results of the RTT measurements to verify the locations. For example, we locate five geo-distributed data centers inside the U.S. of OneDrive based on the RTT measurement results from three North American vantage points although there are no information of these IP addresses in any online databases. After locating all IP addresses, we analyze the five CCSs' load-balancing strategies, respectively. Such knowledge is essential to understand the underlying system designs of CCSs' Web APIs, thus enabling further analysis of API performance.

IV. CCS API SYSTEM DESIGN

The system designs (*i.e.*, API protocol, server location, and load-balancing strategy) are of great importance to CCS Web APIs, with both policy [5] and performance implications. To identify how different CCSs serve their Web APIs, we start by analyzing the application layer protocols of their Web APIs. Then we explore the five CCSs' server locations as well as their different load-balancing strategies.

A. Various CCS API Protocols

In order to protect the privacy of users' data, all the five CCSs utilize standard OAuth protocol for users to authorize third-party applications.

Table II and Table III show the protocols of `upload` and `download` APIs of the five CCSs, respectively. Note that most CCSs use SSL-secured connections for their APIs to ensure the security of user data, whereas DBank's APIs are all based on plain TCP connection. In addition, all CCSs but DBank use one HTTP request in upload APIs, *i.e.*, issuing an HTTP request containing the file content to the specific service domain which returns the file metadata on success. In contrast, DBank requires three request/response exchanges. Similar situation also exists for the `download` APIs that all CCSs but Dropbox need an additional request to get a temporary download URL for each individual file, while Dropbox use a fixed domain name for downloading every file.

Different from data APIs, the control APIs of each CCS are all served in a single service domain (see Table IV), and most APIs only require one request/response exchange.

B. Different Load-balancing Strategies

A total number of 947 unique IP addresses were collected during our measurements. We identify the corresponding geo-location of each IP address based on a hybrid methodology (*i.e.*, online databases and RTT measurement). Fig. 2 shows the identified data and control API server locations of four CCSs [6]. We reveal the load-balancing strategies of the five CCSs' Web APIs below:

Dropbox: Centralized server location. Dropbox hosts its service for data and control APIs in Amazon's North Virginia and Northern California data centers, respectively. The result is consistent with that for its native client [1]. Specifically, the data API servers are addressed by 575 IP addresses in our measurement while that of control API servers is 13.

[4]We use TCP instead of UDP or ICMP since 1) CCS APIs all use TCP service, and 2) many routers, firewalls, and even the server itself drop ICMP packets [7].

[5]Web APIs of Dropbox and Google Drive are unavailable in mainland China due to policy issues.

[6]Since Google Drive adopts a very different system design (see https://peering.google.com/about/delivery_ecosystem.html) from others, we can only locate the edge Points of Presence of Google's private backbone network instead of the data centers.

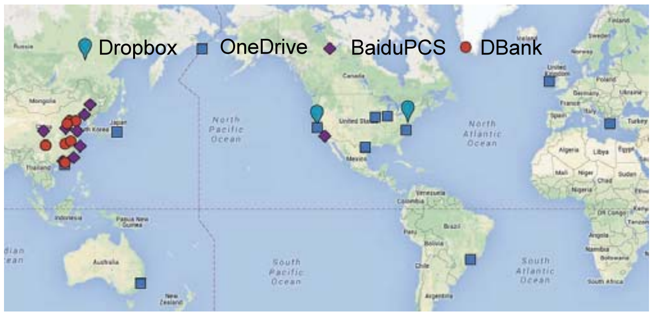


Fig. 2. CCS Web API server locations.

TABLE V
ONEDRIVE'S DATA CENTERS

API	Data centers
Upload, Control	Boydton, VA; Chicago, IL; Des Moines, IA; San Jose, CA; San Antonio, TX
Download	Boydton, VA; Hong Kong; Ireland; Greece; Sao Paolo, Brazil; Saitama, Japan; New South Wales, Australia

OneDrive: Globally distributed data centers. Different from Dropbox, OneDrive uses different sets of IPs for different APIs. Specifically, OneDrive's upload and control APIs rely on five distributed data centers in the U.S. (see Table V), and each API call is randomly served by one of the five data centers regardless of user location. In contrast, OneDrive tries to serve download API calls as close to users as possible based on its globally distributed data centers (see Table V). Remarkably, we observe that not all download API calls are served at the nearest data centers from the vantage points. For example, the Athens node may download files from data centers in Greece or the U.S.. The possible reason may be the delay of data transmission among data centers or load-balancing purposes.

Google Drive: Content delivery ecosystem. Instead of using the public Internet service, Google Drive carries traffic as far as possible on Google's private global backbone network [7]. Specifically, users' traffic will be routed to the closest edge Point of Presence (PoP), from where the traffic goes to Google's data centers via its own network. Since the TCP session is terminated at Google's edge PoPs, the RTT measurement can only reflect the network condition between the vantage points and the edge PoPs, in which case the RTT is generally lower than 10 ms.

BaiduPCS: Three-stage acceleration. BaiduPCS leverages geo-distributed servers for its Web APIs. While the control APIs are served only at Beijing, China, its upload API relies on two locations (*i.e.*, Beijing and Qingdao, China). In addition, observation on the domain names in the temporary URLs used for downloading file content (see Section IV-A) shows a three-stage acceleration mechanism of the download API:

- At first, files can only be downloaded from the servers they were uploaded to.
- Files will be moved to cache servers after they have been downloaded by 50 times. The cache servers are located at Zhejiang, China.
- After certain times of download (50 for files smaller than 2 MB and 20 for others) since cached, the file will be eventually moved to BaiduPCS's CDN servers, which are located at multiple locations, *e.g.*, Zhejiang and Guangdong in China. It is worth noting that we observe some IP addresses located at Los Angeles, U.S., which is the only location outside mainland China.

[7]https://peering.google.com/about/delivery_ecosystem.html

DBank: Meshed system design. DBank's control API relies on its servers at Beijing, China. Considering data APIs, DBank's servers at Hong Kong serve all users outside mainland China, which is a centralized system design like Dropbox. While for users inside mainland China, files will be copied to geo-distributed servers for lower network latency.

V. EVALUATING DATA API

Data storage is the most basic and important functionality provided by CCS Web APIs. It can be drastically affected by the underlying system design. In this section, we first evaluate the overall experience of five CCSs' data APIs. Then we discuss several underlying influencing factors, illustrating the performance implications of various system design choices. Our results provide practical guidance for providers to improve the performance of the APIs, for developers to efficiently use these APIs, and for users to choose appropriate services.

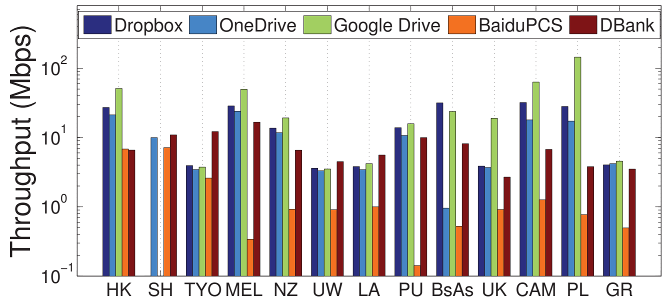
A. Overall Experience

Fig. 3 shows the average throughput of uploading and downloading an 50 MB file to and from the five CCSs on the 13 vantage points (see Fig. 1 for the node names), and Fig. 4 shows the average delay of uploading and downloading an 8 MB file [8]. From the figures we can see that:

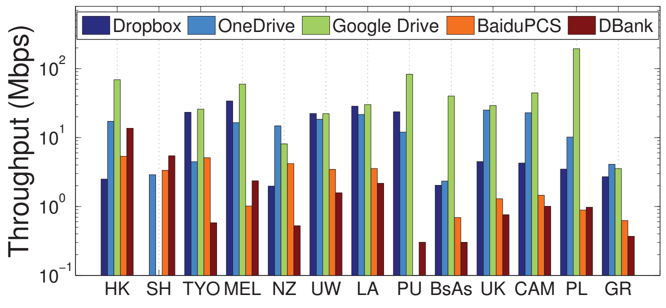
- Considering Dropbox's data API, the performance for transfer of smaller files of MBs is much more unstable across geo-distributed locations compared with that of bigger files (*e.g.*, 50 MB or more). Specifically, the average delay of downloading an 8 MB file from Dropbox on NZ node (Wellington) is about 23 times of that on PU node (Princeton) as evidenced in Fig. 4(b). In the meantime, from Fig. 3(b) we can see that the average throughput of downloading an 50 MB file from Dropbox on PU node is only 12 times of that on NZ node.
- The average upload throughputs of Dropbox and OneDrive are similar at most locations (see Fig. 3(a)). While on quite a few nodes the average download throughputs of OneDrive far exceed that of Dropbox (see Fig. 3(b)), including HK node (Hong Kong), NZ node (Wellington), UK node (St Andrews), CAM node (Cambridge), and PL node (Warsaw).
- While Google Drive outperforms others at most vantage points, CCSs in mainland China (*i.e.*, BaiduPCS and DBank) underperform across different locations.

The following subsections reveal the underlying causes of these observations in terms of different system design choices.

[8]Note that the data APIs offered by Dropbox and Google Drive are unavailable on Shanghai node during our measurement.

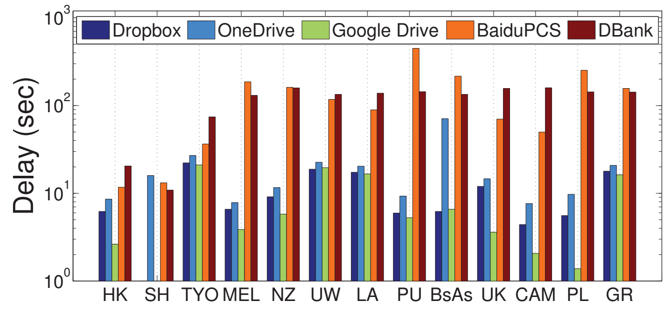


(a) Upload

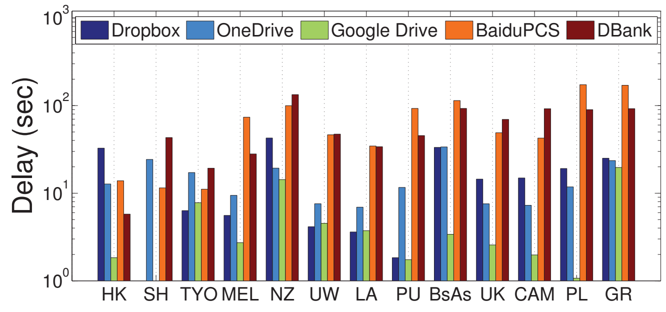


(b) Download

Fig. 3. Average throughput of uploading/downloading a 50 MB file to/from the five CCSs on the 13 PlanetLab nodes.



(a) Upload



(b) Download

Fig. 4. Average delay of uploading/downloading an 8 MB file to/from the five CCSs on the 13 PlanetLab nodes.

B. RTT issue of Dropbox

As mentioned above, the performance of Dropbox's data API becomes vulnerable with small files (*e.g.*, several megabytes). Since most CCS users transfer files of MBs [1], this bottleneck can significantly compromise user experience.

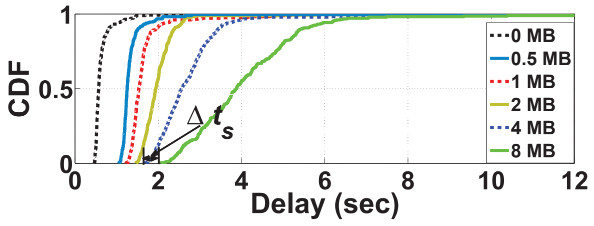
In order to reveal the underlying reason behind the performance fluctuation of Dropbox's data API, we take the measurement data of Dropbox's download API on Seattle node and Melbourne node as an example to give an analysis.

Specifically, the delay of Dropbox's download API consists of the time for:

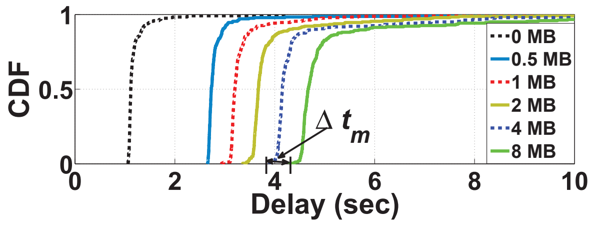
- Session maintenance, including DNS query (nearly constant time), TCP handshake (one RTT), and SSL handshake (two RTTs) before the file transmission, and receiving response message after the file transmission. For comparison, we measure the delay of downloading a 0 MB file as the time for session maintenance.
- File transmission, *i.e.*, the time of transferring the HTTP request message containing the file content.

Fit. 5 shows the CDF of delay of downloading different sized files from Dropbox on Seattle node and Melbourne node. Considering the minimum delay of each file size [9], we can see that when downloading an 8 MB file, the time for transferring the last 4 MB data (*i.e.*, Δt_s in Fig. 5(a) and Δt_m in Fig. 5(b)) is almost the same, while the time of session maintenance (the delay of 0 MB file) and transferring the first 4 MB data lead to the huge performance gap between the two nodes.

[9]We focus on the best achievable performance in this subsection to eliminate the interference of network congestion.



(a) Seattle node



(b) Melbourne node

Fig. 5. CDF of delay of downloading different sized files from Dropbox on Seattle node and Melbourne node.

The reason comes from the RTT issue of Dropbox's centralized system design. Specifically, in our measurement, the minimum RTTs are about 67 ms on Seattle node and 224 ms on Melbourne node, respectively. Since these HTTP implementations all take TCP as the transport-layer connection protocol, the large RTT can not only extend the time of TCP and SSL handshake, but also prolong the TCP slow start phase. For illustration, we compute the amounts of data the TCP flows need to transfer to achieve the maximum throughput on the two vantage points as in [8]. The results are 1.9 MB on Seattle node and 6.3 MB on Melbourne node, respectively, which are similar to our measurement data as showed in Fig. 5.

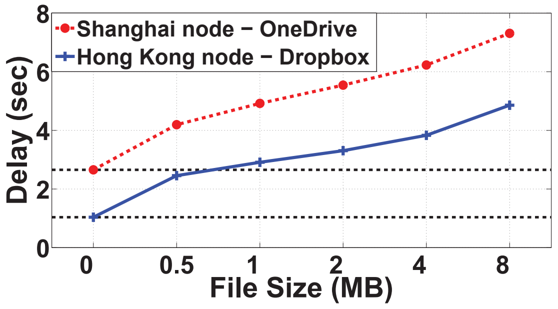


Fig. 6. Minimum delay of downloading different sized files from OneDrive on Shanghai node, and that from Dropbox on Hong Kong node.

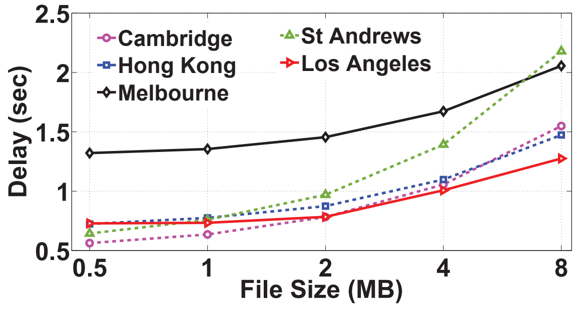


Fig. 7. Minimum delay of downloading different sized files from Google Drive on five geo-distributed PlanetLab nodes.

This indicates that the slow start phase prolonged by the large RTT is the main reason of the performance depreciation of Dropbox's data API on Melbourne node.

The same problem also exists on other vantage points in our measurement, especially for those outside the U.S.. In fact, according to the RTT measurement, the RTTs to Dropbox's file servers varies from 23 ms (on Princeton node) to 224 ms (on Melbourne node). The diverse RTTs lead to the performance fluctuation of Dropbox's data API, especially when transferring files of MBs. This suggests that third-party application developers should keep the underlying TCP connection alive longer in order to reduce slow start times so as to maintain a high throughput.

C. Geo-distributed Data Centers of OneDrive

We observe from Fig. 3 that although the average upload throughput of OneDrive is similar to that of Dropbox on a global scale, OneDrive's download API outperforms that of Dropbox on plenty of vantage points, including HK node (Hong Kong), NZ node (Wellington), UK node (St Andrews), CAM node (Cambridge), and PL node (Warsaw).

The reason is that OneDrive's Web APIs rely on its globally distributed data centers to reduce RTT between users and servers, especially for its download API (see Section IV-B). Specifically, while OneDrive's upload API relies on five data centers in the U.S., users of OneDrive outside the U.S. may download files from a closer data center compared with users of Dropbox.

The geo-distributed system design, however, brings drawbacks with benefits. Since the file requested by a user may be stored in several data centers, an additional request is necessary in order to get the current download URL of the file (see Section IV-A). The complex protocol may introduce non-negligible performance overhead in certain cases.

As an example, Fig. 6 shows the minimum delay of downloading different sized files from OneDrive on Shanghai node, and that from Dropbox on Hong Kong node. We can observe that Dropbox outperforms OneDrive in terms of each file size because of the huge gap between the delays of downloading a 0 MB file (*i.e.*, the time of session maintenance as explained in Section V-B) on the two nodes. Since the RTTs in the two datasets are almost the same (*i.e.*, 222 ms [10] and 220 ms, respectively), we can infer that the protocol overhead is the main reason that depreciate the performance of OneDrive.

As mentioned in Section IV-A, the protocol overhead problem exists for all the CCSs except Dropbox, since only Dropbox adopts a centralized system design. This suggests that CCS providers should try to adopt a simpler protocol in order to improve the quality of their API services.

D. Content Delivery Ecosystem of Google Drive

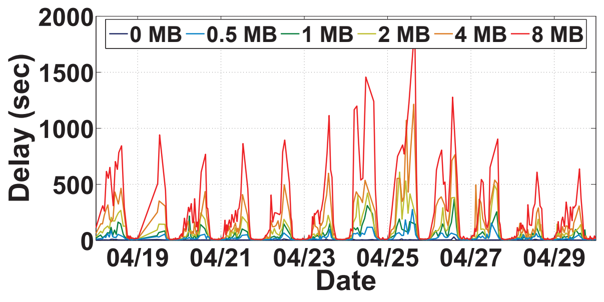
Although we can not measure the RTTs between the vantage points and Google Drive's file servers directly as mentioned in Section IV-B, we observe significantly low RTTs from the measurement data of Google Drive's data API. For illustration, we plot the minimum delay of downloading different sized files from Google Drive on five vantage points in Fig. 7. We can observe that the time of file transmission (as defined in Section V-B) grows exponentially with file size doubles on each node, which implies that the TCP flows have reach the maximum throughput after sending little amounts of data. In fact, the data needed to be sent to achieve the maximum throughput is generally lesser than 1 MB on the 13 vantage points in our measurement. We can infer that the RTTs between these vantage points and Google Drive's file servers are extremely low.

The reason is that Google Drive's Web APIs relies on its content delivery ecosystem instead of public Internet service (see Section IV-B), which is a very different design choice compared with the other CCSs. As a consequence, the low RTTs between users and Google Drive's file servers lead to the excellent performance of Google Drive across locations.

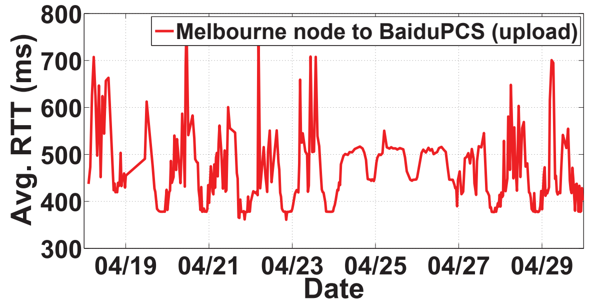
E. Network Congestion of BaiduPCS and DBank

In our measurement, BaiduPCS and DBank underperform others at most locations. The reason comes from the severe network congestion. As an example, Fig. 8(a) shows the delay of uploading files to BaiduPCS on Melbourne node from April 18, 2015 to April 29, 2015, and Fig. 8(b) plots the corresponding RTT during the same period. Since network congestion can be reflected by the prolonged RTT, we can observe that the data API performance is significantly depreciated by network

[10] In our measurement, we observe that Shanghai node always download files from OneDrive's data centers inside the U.S..



(a) Delay



(b) Average RTT

Fig. 8. Delay of uploading different sized files to BaiduPCS on Melbourne node and the corresponding average RTT.

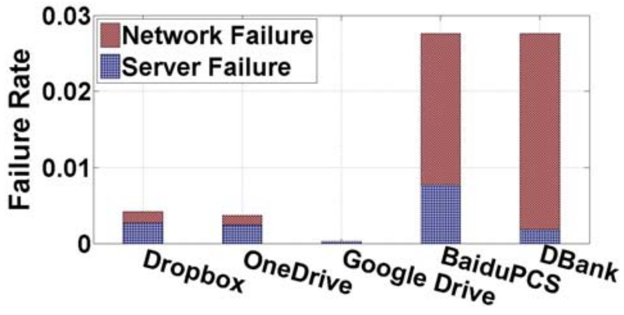


Fig. 9. failure rates of the five CCSs' data APIs.

congestion. In our measurement, the RTT to BaiduPCS and DBank is much more volatile than to others across the 13 nodes, which implies that network congestion is the main bottleneck of the data API of BaiduPCS and DBank.

However, we can observe from Fig. 3 and Fig. 4 that on certain vantage points such as SH node (Shanghai), BaiduPCS and DBank provide better performance than others. The observation suggests that users should choose appropriate CCS providers according to their geographical locations.

F. Data API Failure Rate

Failure rate of data API is crucially important for the perceived experience of upper-layer applications. Fig. 9 shows the failure rates of the five CCSs' data APIs during our measurement. Specifically, we classify all failure into two groups, *i.e.*, server failure (including HTTP 5XX errors) and network failure (including connect errors, read errors, and write errors). We can see that the failure rates of BaiduPCS and DBank are much higher than others with vast amounts of network failure, which is the consequence of the network

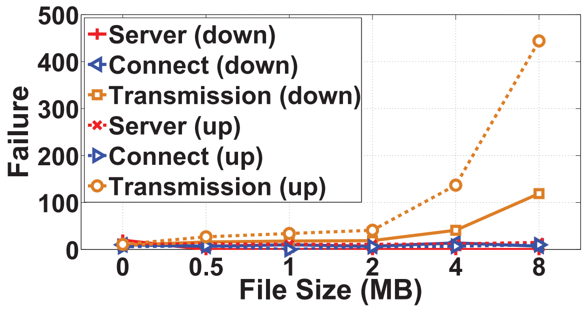


Fig. 10. # of failed API calls of different sized files of DBank.

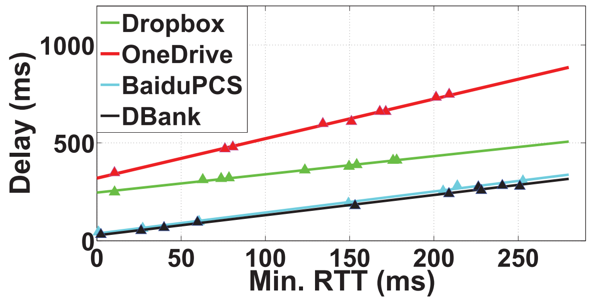


Fig. 11. Delay versus RTT of the `delete folder` APIs of four CCSs.

congestion problem. In contrast, the failure rate of Google Drive is significantly lower than others because of its private backbone network.

Furthermore, we take DBank, which has the highest failure rate of the five CCSs, as an example, to reveal the correlation between failure rate and file size (see Fig. 10). Note that we further classify network failure into two groups, *i.e.*, connection failures (*i.e.*, connect errors) and transmission failures (*i.e.*, read errors and write errors). As a result, we can see that the transmission failure rate increases significantly with file size. In conclusion, third-party applications should transfer large files in multiple trunks via advanced data APIs (see Section IV-A) to minimize losses of failed data API calls.

VI. Evaluating Control API

Control APIs are essential for building full-fledged CCS applications. In this section, we study the performance of the five CCSs' control APIs, and reveal the major impact factors.

A. Server Processing Time

In practice, multiple control APIs are always called in sequence via a single connection. Accordingly, we only measure the time of transferring HTTP messages without the time for connection establishment in the control API measurement. From the measurement data we observe a linear relation between the delay of control APIs and RTT. As an example, Fig. 11 shows the delay versus RTT of the `delete folder` APIs of four CCSs [11]. The reason is that the messages of control APIs are generally very short. As a consequence, the delay of a control API call consists of a small and constant

[11]We do not include Google Drive here since we can only measure the RTT to its edge PoPs instead of where the requests are actually served.

TABLE VI
SERVER PROCESSING TIME (MILLISECOND) OF CONTROL APIS

API	Dropbox	OneDrive	BaiduPCS	DBank
Create folder	204.40	457.06 (2)	59.02	31.37
Move file	250.35	678.28 (3)	51.85	39.67
Move folder	257.46	641.01 (3)	53.63	47.90
Copy file	309.61	802.16 (3)	72.42	41.44
Copy folder	265.02	—	50.35	48.64
Delete file	219.24	340.91 (2)	49.69	28.32
Delete folder	246.18	319.51 (2)	36.71	27.56
List	79.55	138.46 (1)	65.68	20.12
Share	87.11	540.07 (2)	—	—

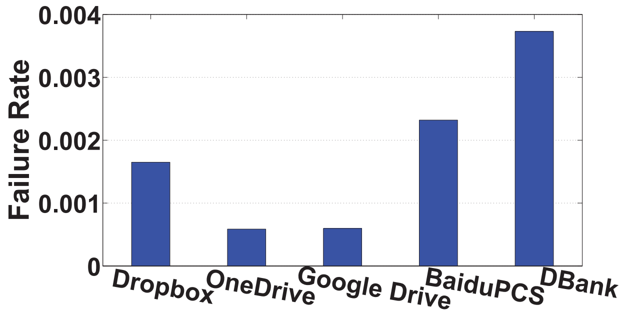


Fig. 12. Failure rates of the five CCSs' control APIs.

number of RTTs (see the slope of lines in Fig. 11) and a constant time for server processing (see the offset on y-axis).

By calculating the linear regression of delay on RTT, we summarize the server processing time of each control API in Table VI. Note that OneDrive occasionally need additional requests to query file/folder IDs, and we record the number of HTTP requests needed by each OneDrive's Control API in parentheses. We can see that the control APIs of BaiduPCS and DBank outperform that of Dropbox and OneDrive, and OneDrive has a server processing time of up to 802 ms. Our results in Table VI provide a reference for predicting the performance of control APIs from the RTT. In addition, third-party applications are strongly recommended to maintain a local cache of file/folder metadata to reduce the overhead caused by the complex API protocol of OneDrive.

B. Control API Failure Rate

Different from data APIs, the failure rate of control APIs is much lower. Fig. 12 shows the overall failure rate of each CCS's control API. We can see that even for DBank, which performs the worst in our global measurement, the control API failure rate is less than 0.4%, and those of Google Drive and OneDrive are less than 0.06%.

VII. RELATED WORK

Cloud-based services are now getting pervasive on a large scale. Many prior works have focused on distinct aspects of cloud services, such as cloud service pricing policy [9], datacenter carbon emission [10], datacenter network management [11], [12], and datacenter virtual machine [13].

Cloud storage, as one of the most important components of cloud services, has gained increasing momentum within research community. A number of previous works study cloud storage from different perspectives, including the design and implementation of the service infrastructure [14], [15], [16], [17], privacy and security issues [18], [19], [20], [21], [22], and characteristics and access patterns of the data residing in cloud storage systems [23]. In addition, some studies focus on current public offerings [24]. Specifically, the performance of Microsoft Azure is explored in [25], Amazon S3 is measured in [26] to determine if cloud storage is suitable for scientific grids, and a client-side performance analysis of Amazon S3 is made in [27].

Unlike cloud storage like Amazon S3 and Windows Azure Storage, CCS provide services for saving personal files, synchronizing devices and sharing content with others. As the earliest and most popular CCS, Dropbox has been studied extensively from different angles. Through measurements on a university campus and in residential networks, Drago *et al.* characterize Dropbox and its user workload, and analyzes system architecture and storage protocol to reveal possible performance bottlenecks [1]. Similarly, Wang *et al.* [2] and Li *et al.* [3] note the existence of performance bottlenecks in Dropbox-like cloud storage services, and propose new mechanisms to overcome them. In addition, others characterized Dropbox client workloads [28], Quality of Experience (QoE) of Dropbox-like services [29], and the possibility of unauthorized data access and the security implications of storing data in Dropbox [30]. Some works also study other popular CCS solutions like Wuala [31]. In addition, Tang *et al.* synergize multiple CCSs to improve performance, reliability, and security [32].

Instead of focusing on a specific service, plenty of studies compare alternative CCS providers. Specifically, Drago *et al.* compare system architecture and synchronization performance of five CCS services [4]. Hu *et al.* study the backup and restore performance as well as privacy related issues of four CCS services by comparing their traffic usage, delay time, and CPU usage of uploading new files [5]. Li *et al.* extend their previous work [3] to unravel the general factors that may significantly affect the data sync traffic [6]. Different from these studies, our work presents a long-term global measurement of five popular CCSs to reveal the diverse and varying characteristics in terms of system designs as well as the performance implications of different design choices.

Above works all focus on native clients of CCS providers. However, their RESTful Web APIs are very different from native clients in purpose, capability, and system design. Only a few have studied the Web APIs. An active measurement study of three CCSs' Web APIs is presented in [33], providing statistical distributions that model various key performance aspects to characterize their QoS of Web APIs, such as transfer speed and failure rate, but does not investigate the underlying system designs of these services.

Our work is different from previous studies by revealing the underlying system designs of five popular CCSs' Web APIs, and analyzing the performance implication of different system design choices. Through a long-term global measurement study, we provide practical guidance for service providers to optimize their API performance, for developers to improve the experience of third-party applications, and for users to pick appropriate services that best match their requirements.

VIII. Conclusion

In this work, we perform a comprehensive empirical study of the RESTful Web APIs of five popular CCSs. We develop a set of measurement tools, and deploy a trial on 13 geo-distributed PlanetLab nodes covering 13 countries across 5 continents for 20 consecutive days in April, 2015. Our results uncover the system designs of the five CCSs' Web APIs, and reveal their diverse and varying performance in both spatial and temporal dimensions.

Our analysis leads to six findings: 1) The five CCSs have very different system designs. We observe that only Dropbox adopts a centralized system design, and Google Drive relies its service on the content delivery ecosystem. 2) Since Dropbox hosts its data APIs only in Amazon's North Virginia data center, the performance is depreciated by RTT issue on many locations. Third-party developers should keep the underlying TCP connection alive longer in order to maintain a high throughput. 3) Although geo-distributed system design improves the performance of data APIs, the complex protocol may introduce non-negligible performance overhead in certain cases. Service providers should try to adopt a simpler protocol in order to improve the quality of service. 4) The performance of BaiduPCS and DBank is limited outside the mainland China as a consequence of severe network congestion. While Dropbox and Google Drive are unavailable inside mainland China. Thus users should choose appropriate CCS providers according to their geographical locations. 5) The failure rate of data API increases significantly with file size. Third-party developers should transfer large files in multiple trunks to minimize losses of failed data API calls. 6) Server processing time has significant impact on the delay of control APIs. Developers should take this into consideration and design appropriate business logic for interactive applications.

References

[1] I. Drago, M. Mellia, M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding Personal Cloud Storage Services," in *Proc. of IMC*. ACM, 2012.

[2] H. Wang, R. Shea, F. Wang, and J. Liu, "On the Impact of Virtualization on Dropbox-like Cloud File Storage/Synchronization Services," in *Proc. of IWQos*. IEEE, 2012.

[3] Z. Li, C. Wilson, Z. Jiang, Y. Liu *et al.*, "Efficient Batched Synchronization in Dropbox-Like Cloud Storage Services," in *Proc. of Middleware*. ACM, 2013.

[4] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking Personal Cloud Storage," in *Proc. of IMC*. ACM, 2013.

[5] W. Hu, T. Yang, and J. N. Matthews, "The Good, the Bad and the Ugly of Consumer Cloud Storage," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, 2010.

[6] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu *et al.*, "Towards Network-level Efficiency for Cloud Storage Services," in *Proc. of IMC*. ACM, 2014.

[7] K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, and D. Wetherall, "Improving the Reliability of Internet Paths with One-hop Source Routing," in *Proc. of OSDI*. USENIX, 2004.

[8] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proc. of INFOCOM*. IEEE, 2000.

[9] X. Yi, F. Liu, Z. Li, and H. Jin, "Flexible Instance: Meeting Deadlines of Delay Tolerant Jobs in The Cloud with Dynamic Pricing," in *Proc. of ICDCS*. IEEE, 2016.

[10] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin, "Carbon-aware Online Control of Geo-distributed Cloud Services," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, 2015.

[11] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN Controller Assignment in Data Center Networks: Stable Matching with Transfers," in *Proc. of INFOCOM*. IEEE, 2016.

[12] J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair Network Bandwidth Allocation in IaaS Datacenters via a Cooperative Game Approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, 2016.

[13] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions," *Proceedings of the IEEE*, vol. 102, no. 1, 2014.

[14] B. Calder, J. Wang *et al.*, "Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency," in *Proc. of SOSP*. ACM, 2011.

[15] M. Vrable, S. Savage, and G. Voelker, "Cumulus: Filesystem Backup to the Cloud," in *Proc. of FAST*. USENIX, 2009.

[16] ——, "BlueSky: A Cloud-backed File System for the Enterprise," in *Proc. of FAST*. USENIX, 2012.

[17] Y. Huang, Z. Li, G. Liu, and Y. Dai, "Cloud Download: Using Cloud Utilities to Achieve High-quality Content Distribution for Unpopular Videos," in *Proc. of MM*. ACM, 2011.

[18] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-of-Clouds," *Trans. Storage*, vol. 9, no. 4, 2013.

[19] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, "Depot: Cloud Storage with Minimal Trust," in *Proc. of OSDI*. USENIX, 2010.

[20] D. Kholia and P. Wegrzyn, "Looking Inside the (Drop) Box," in *Proc. of WOOT*. USENIX, 2013.

[21] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of Ownership in Remote Storage Systems," in *Proc. of CCS*. ACM, 2011.

[22] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side Channels in Cloud Services: Deduplication in Cloud Storage," *IEEE Security and Privacy*, vol. 8, no. 6, 2010.

[23] S. Liu, X. Huang, H. Fu, and G. Yang, "Understanding Data Characteristics and Access Patterns in a Cloud Storage System," in *Proc. of CCGrid*. IEEE, 2013.

[24] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of Backup Workloads in Production Systems," in *Proc. of FAST*. USENIX, 2012.

[25] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early Observations on the Performance of Windows Azure," in *Proc. of HPDC*. ACM, 2010.

[26] M. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for Science Grids: A Viable Solution?" in *Proc. of DADC*. ACM, 2008.

[27] A. Bergen, Y. Coady, and R. McGeer, "Client Bandwidth: The Forgotten Metric of Online Storage Providers," in *Proc. of PacRim*. IEEE, 2011.

[28] G. Goncalves, I. Drago, A. Couto da Silva, A. Borges Vieira *et al.*, "Modeling the Dropbox Client Behavior," in *Proc. of ICC*. IEEE, 2014.

[29] P. Casas, H. Fischer, S. Suette, and R. Schatz, "A First Look at Quality of Experience in Personal Cloud Storage Services," in *Proc. of ICC*. IEEE, 2013.

[30] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, "Dark Clouds on the Horizon: Using Cloud Storage As Attack Vector and Online Slack Space," in *Proc. of SEC*. USENIX, 2011.

[31] T. Mager, E. Biersack, and P. Michiardi, "A Measurement Study of the Wuala On-line Storage Service," in *Proc. of P2P*. IEEE, 2012.

[32] H. Tang, F. Liu, J. Shen, Y. Jin, and C. Guo, "UniDrive: Synergize Multiple Consumer Cloud Storage Services," in *Proc. of Middleware*. ACM, 2015.

[33] R. Gracia-Tinedo, M. Artigas, A. Moreno-Martinez, C. Cotes, and P. Lopez, "Actively Measuring Personal Cloud Storage," in *Proc. of CLOUD*. IEEE, 2013.