An Empirical Analysis of a Large-scale Mobile Cloud Storage Service

Zhenyu Li†, Xiaohui Wang†, Ningjing Huang†, Mohamed Ali Kaafar‡,
Zhenhua Li⋆, Jianer Zhou¶, Gaogang Xie†, Peter Steenkiste♯

†ICT-CAS, ‡CSIRO Data61, ⋆Tsinghua Uni., ¶CNIC-CAS, ♯CMU

{zyli, wangxiaohui, huangningjing, xie}@ict.ac.cn, dali.kaafar@nicta.com.au,
lizhenhua1983@tsinghua.edu.cn, zhoujianer@cnic.cn, prs@cs.cmu.edu

ABSTRACT

Cloud storage services are serving a rapidly increasing number of mobile users. However, little is known about the differences between mobile and traditional cloud storage services at scale. In order to understand mobile user access behavior, we analyzed a dataset of 350 million HTTP request logs from a large-scale mobile cloud storage service. This paper presents our results and discusses the implications for system design and network performance. Our key observation is that the examined mobile cloud storage service is dominated by uploads, and the vast majority of users rarely retrieve their uploads during the one-week observation period. In other words, mobile users lean towards the usage of cloud storage for backup. This suggests that delta encoding and chunk-level deduplication found in traditional cloud storage services can be reasonably omitted in mobile scenarios. We also observed that the long idle time between chunk transmissions by Android clients should be shortened since they cause significant performance degradation due to the restart of TCP slow-start. Other observations related to session characteristics, load distribution, user behavior and engagement, and network performance.

Keywords

Mobile cloud storage; user behavior; TCP performance

1. INTRODUCTION

With the abundant and pervasive personal content generation witnessed today, the use of cloud storage for storing and sharing personal data remotely is increasing rapidly. The personal cloud storage market is estimated to have a compound annual growth rate of 33.1% between 2015 and 2020 [5]. Major players such as Google, Microsoft, Apple, Baidu, and Dropbox are competing to offer users the best quality of service while keeping their costs low. Cloud storage providers are working hard to meet their growing number of mobile users. For instance, Dropbox redesigned its mobile app, adding new functionality, and it tapped mobile ISPs for improved user experience [4].

Both improving user experience and keeping costs low can benefit from an in-depth understanding of the following two system aspects:

- *User behavior pattern*, including workload variation, session characteristics, and usage patterns (e.g., occasional *vs.* heavy use). Insight gained can help optimize performance and reduce cost at both the server and client.
- *Data transmission performance*, where factors that increase latency must be identified and addressed to improve user QoE (Quality of Experience), a key factor in user loyalty.

Unfortunately, little is known about these two system aspects in *mobile* cloud storage services. Recent seminal work examined user behavior in Dropbox [12, 8] and Ubuntu One [16]. Other research focuses on the traffic inefficiencies resulting from the synchronization of frequent local changes [21, 22]. However, all these studies are specific to traditional PC clients, rather than mobile users. While some work noted the unique challenges of synchronizing data of mobile users in cloud storage services [10], both the user access behavior and network performance properties in *large-scale* mobile cloud storage services remain unexplored. Indeed, mobile users might behave quite differently from PC-based users when using cloud storage services. For instance, mobile users may modify file content less often due to the inconvenience of editing files on mobile terminals. Also, the properties of mobile devices might limit transmission performance due to connectivity, power and even software constraints.

This paper fills this void by examining a unique dataset collected from a large-scale mobile cloud storage service

IMC 2016, November 14-16, 2016, Santa Monica, CA, USA

DOI: http://dx.doi.org/10.1145/2987443.2987465

for a one-week period. The dataset consists of 349 million HTTP request logs generated by 1.1 million unique *mobile users*. Our analysis first separates the series of requests of each user into more fine-grained sessions with the session interval empirically learned from the dataset. We then characterize session attributes and usage patterns. Finally, we examine data transmission performance and diagnose the performance bottlenecks using packet-level traces collected at both the server and client sides. Our main findings and their implications are as follows:

- *Sessions and burstiness.* The inter-file operation time of individual users follows a two-component Gaussian mixture model, where one component captures the in-session intervals (mean of 10s), and the other corresponds to the inter-session intervals (mean of 1 day). This model allows us to characterize user behavior at session level. For example, users store and retrieve files in a bursty way, as they tend to perform all file operations within a short time period at the beginning of sessions, and then wait for data transmission to finish.
- *Storage-dominated access behavior.* In a single session, mobile users either only store files (68.2% of sessions), or only retrieve files (29.9% of sessions), but they rarely perform both. The mixture-exponential distribution model for average file size reveals that 91% of the storage sessions store files that are around 1.5 MB in size. Surprisingly, over half of the mobile users are predominantly interested in the cloud storage for backing up their personal data, and they seldom retrieve data. In contrast, PC-based users are far more likely to fully exploit both storage and retrieval processes.
- *Distinct engagement models.* User engagement exhibits a bimodal distribution, where users either return soon to the service or remain inactive for over one week. Notably, about 80% of the users that use multiple mobile terminals will not return to download their uploads in the following week. On the other hand, users that use both mobile and PC clients account for 14.3% of users, and they are more likely to retrieve their uploads very soon.
- *Device type effects on performance.* The mobile device type (either Android or iOS) has a significant impact on chunk-level transmission performance. We show that the root cause lies in the idle time between chunk transmissions. A large TCP idle time might trigger the restart of TCP slow-start. As a result, as many as 60% of idle intervals on Android devices restart TCP slow-start for the next chunk, compared with only 18% for iOS. Moreover, the small TCP receive window size advertised by servers limits upload performance, independent of device type.

Our results show that mobile users tend to use the cloud storage service for backup of personal data. This suggests opportunities for simplifying the design, and for optimizing the performance of mobile cloud storage services. In particular, expensive delta encoding and chunk-level deduplication implemented in traditional cloud storage services [21, 22] have limited benefits in mobile scenarios. Instead, we posit a smart `auto backup` function that defers uploads in order to reduce the peak load and reduce cost. Some cost-effective storage solutions for infrequently accessed objects, like f4 [26], can also easily reduce cost. In addition, providers can leverage the distinct usage patterns of users for effective in-app advertisement. Finally, to improve network, the effect of long idle intervals between chunks should be mitigated by using larger chunks or batching the transmission of multiple chunks.

Another contribution of this work is that we have made our dataset publicly available for the community at *http://fi.ict.ac.cn/data/cloud.html*. We hope the dataset will be used by other researchers to further validate and investigate usage patterns in mobile cloud storage services.

The remainder of this paper is organized as follows. Section 2 describes the background and dataset, while Section 3 presents an in-depth analysis of user behavior. Section 4 investigates data transmission performance. We discuss the caveats of our analysis in Section 5, and related work in Section 6. Finally, Section 7 concludes our work.

2. DATASET AND WORKLOAD

This section begins with an overview of the mobile cloud storage service that we examine in this paper, followed by the description of the dataset in use and its limitations. Finally, we present the temporal pattern of workload from mobile devices, which will facilitate the understanding of system artifacts.

2.1 Overview of the Examined Service

The service that we examined is one of the major personal cloud storage services in China, serving millions of active users per day. The service is very similar to other state-of-the-art cloud storage services, like Google Drive and Microsoft OneDrive. Users are offered to *store, retrieve, delete and share* [1] files through either a PC client or a mobile app. This paper focuses on the first two operations (i.e., store and retrieve), because the last two operations (i.e., delete and share) do not go through the storage front-end servers, where we collected our data.

Users are allowed to select multiple files to store or retrieve at one time. Uploading a file does not automatically delete the local copies of the files. That said, a local copy of the uploaded file might be kept on the device. HTTP is used to move data between cloud servers and clients. The basic object for a HTTP request is a chunk of data with a fixed size of 512 KB (except for the last chunk of HTTP content). Each chunk is uniquely identified by an MD5 hash value derived from the chunk data. Files larger than the maximum chunk size are split into chunks for transmission and storage. A file can be identified using either a URL or its MD5 hash value, or both.

[1]Delta updates (also known as direct modifications on files) are currently not supported. In other words, any change to the local copy of a file that results in change of the file's MD5 value will eventually lead to a new file to be uploaded.

The mobile app is available for both iOS and Android devices. To store or retrieve a file, a mobile device first contacts a pre-defined metadata server. For storage, the mobile device sends the file's metadata (i.e., the file name and MD5 hash value) to a metadata server. The metadata server first checks whether or not the file has been already uploaded to a storage server. If storage server has already kept a copy of the file, the metadata server adds the file to the user's space and tells mobile device not to upload the file. This deduplication process aims to avoid redundant uploads of the same content, so as to reduce the workload of storage servers. Otherwise (i.e., if the file version has not been identified on any storage server), the metadata server sends the client the identity of the closest storage front-end server to contact. The mobile device then sends some file information (including the file name, the file size, the file MD5, the number of chunks and the corresponding MD5 values of individual chunks) to the front-end server via a *file storage operation request*, and then initiates the storage process of chunks with *chunk storage requests*.

In the case of a retrieval query, the mobile device asks a pre-defined metadata servers for the MD5 hash value of the requested file (indicated by the unique URL), which is then used to request the file relevant information from a storage server via a *file retrieval operation request*, followed by requests of individual chunks (i.e., *chunk retrieval requests*). For ease of description, file storage/retrieval operation request is denoted as *file operation*, while chunk storage/retrieval request is shortened as *chunk request*.

Users are allowed to store or retrieve multiple files at a time, and several files might be transferred in parallel. For example, the mobile app allows users to backup multiple photos at a time. Users can also issue file storage or retrieval requests when other files are being transferred. For a single file, HTTP requests can use one or more TCP connections. TCP connections can also carry HTTP requests from more than one file. Within a specific TCP connection, chunks are sequentially requested, i.e., a new chunk request will not be issued until the receiver explicitly acknowledges (at the HTTP level) the previous chunk[2].

2.2 Dataset Description

HTTP request logs. We collected HTTP-level request logs from *all* storage front-end servers of the examined mobile storage service. We focus on the logs from mobile devices in this paper. Table 1 lists an example of the main fields contained in a log entry.

The device ID uniquely identifies a device, while the user ID is uniquely bound to a registered user account. Both device ID and user ID are anonymized in our datasets. The data volume measures the volume of uploaded (resp. downloaded) data for a chunk storage (resp. retrieval) request. The request processing time measures the duration between the first bytes received by front-end server and the last bytes sent to mobile client. The average RTT is the average of all RTTs measured for the TCP connection on which the HTTP request is transferred. Finally, whether the request is proxied or not is obtained from the HTTP header `X-FORWARDED-FOR`.

[2]The batched store/retrieve operations [11] of multiple chunks are not yet supported.

Table 1: Main fields of logs.

Field	Example
Timestamp	19:10:01 Aug. 4 2015
Device type	Android or iOS
Device ID	33ab8c95437fd
User ID	1355653977
Request type	file operation/chunk request
Data volume	512 KB
Req. processing time	4.398s
Average RTT	89.238ms
Proxied or not	yes

We collected all the log entries of HTTP requests originated from mobile devices for one week in August 2015. In total, we obtained 349,092,451 logs from 1,148,640 active mobile users (identified by user ID) using 1,396,494 mobile devices (identified by device ID), where 78.4% of the accesses were from Android devices, and the rest from iOS devices. Geographically, users are located in China as well as in overseas countries. A user might use several mobile devices to access the service.

Users can use both mobile devices and PC clients to access the service. In our dataset, there are 164,764 such users, accounting for 14.3%. We also collected the HTTP request logs generated by these users when they accessed via PC clients, corresponding to 59,647,797 logs. Note that since we use complete HTTP request logs (as opposed to sampling the logs), the information of all the used devices (either mobile devices or PC devices) of a user is included in our dataset. In addition, to examine the disparity between mobile users and PC client users, we extract 1,206,592,592 request logs from over 2 million PC-based users during the same period of time. These logs are used for usage pattern analysis in §3.2.

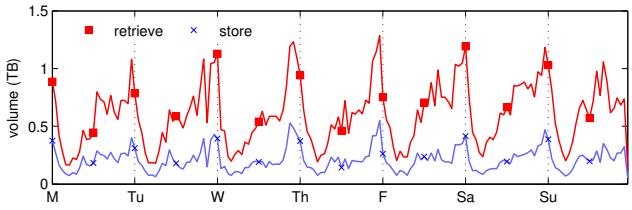
Packet-level traces. We also captured packet-level traces (128-byte packet) for storage and retrieval flows originated from mobile devices at one of the storage front-end servers. The packet-level traces are desired for the investigation of TCP behavior and its impact on the cloud storage service. In total, we obtained the packet-level traces of 40,386 flows, including both storage and retrieval flows.

2.3 Dataset Limitations

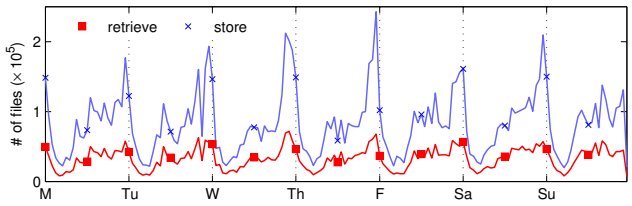
Due to privacy constraints, none of our traces include individual file information or chunk hashes. This prevents us from linking HTTP requests that are associated with the same file. Instead, we group requests into sessions based on the inter-file operation times (See §3.1 for details). A session corresponds to the activities of a user in a period of time prior to a relatively longer inactivity period of time (e.g., logging out). A session can contain multiple file operations.

In addition, we analyze the cloud storage performance by examining the TCP behavior of storage/retrieval flows. As TCP is sender driven, packet traces at client side (i.e., TCP sender when storing data), which are not included in the dataset, are required when analyzing the performance of storage flows. We resort then to a number of active measurements for this purpose (See §4).

2.4 Workload Overview



(a) data volume



(b) # of retrieved/stored files

Figure 1: Temporal variation of workload: (a) the data volume, (b) number of files that are retrieved/stored. The markers represent 12PM and 24PM of each day.

We first report the temporal pattern of the mobile cloud storage workload in Figure 1. Request logs are grouped into one-hour frame bins. For each bin, the total data volume of storage and retrieval, which reflects the load on storage servers, is plotted in Figure 1a, while the number of stored and retrieved files, which reflects the load on metadata servers, is shown in Figure 1b. We observe a clear diurnal pattern with a sharp surge around 11PM when users are at home where they are likely to have access to WiFi. In fact, the mobile app provides users the option to transfer files only via a WiFi network. We can also observe that retrievals contribute more data volume to the workload than storages. In contrast, the number of stored files per hour is over two times of that of retrieved files. This implies that retrieved file objects are much larger than the stored ones, which is also confirmed by our analysis in §3.1.

The above observations on workload variation have two-fold implications. First, both storage servers and metadata servers would be highly over-provisioned for most of the time, since the server capacity is often designed to bear the peak load. Elastic scale-in and scale-out of the service as such are needed to address this over-provision problem. Second, the huge data volume greatly challenges the storage space and bandwidth. In this context, a thorough analysis of the system artifacts (§3) and performance (§4) can shed light on avenues for system and revenue optimizations.

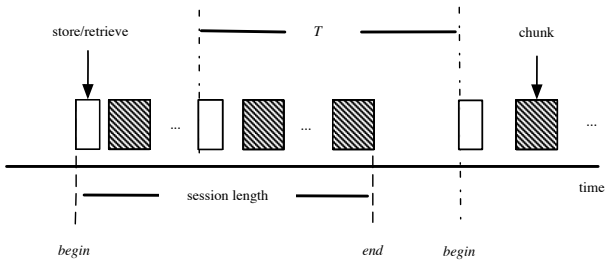


Figure 2: File operation interval and session identification methodology for a user. The white boxes represent file operations, while the hatched boxes show chunk requests.

3. USER BEHAVIOR ANALYSIS

3.1 Session Characteristics

We first examine the inter-file operation time. The analysis leads to the session identification. We then analyze the session size and build models to capture the intrinsic properties of file sizes.

3.1.1 *File Operation Interval and Session Identification*

Our dataset captures a user's activity as a stream of HTTP requests with their associated timestamps as shown in Figure 2. A file operation request, which carries the requested file information to storage front-end servers, corresponds to the beginning of file storage or retrieval (white box in Figure 2). The file operation interval (T in Figure 2) measures the time duration between each file operation request and the previous operation of the same user. We separate sessions based on the distribution of file operation interval. Here, a session is formally defined as a sequence of HTTP requests (including both file operations and chunk requests) made by a user during a period of time, in which the time difference between any two sequential file operations is less than τ, where τ is a parameter that should be empirically derived. In other words, a file operation request begins a new session of the user if it is more than τ away from the previous file operation, i.e., $T > \tau$.

Figure 3 plots the histogram distribution of the file operation intervals for the request streams of all users extracted from our dataset, based on the logarithmically scaled inter-file operation time. We observe a valley around the 1-hour mark. Based on the session identification methodology in [18], this suggests setting τ to one hour. We further fit a two-component Gaussian mixture model, where the expectation maximization (EM) method [7] is used to find the maximum likelihood estimate of the model parameters. As shown in Figure 3, one component corresponds to within-session intervals with an average around 10s. The other captures the inter-session intervals with an average around 1 day, corresponding to the behavior that some users return to the service after a one-day period. This model further confirms that setting τ to 1 hour is reasonable as the 1-hour mark is equally likely to be within the two components.

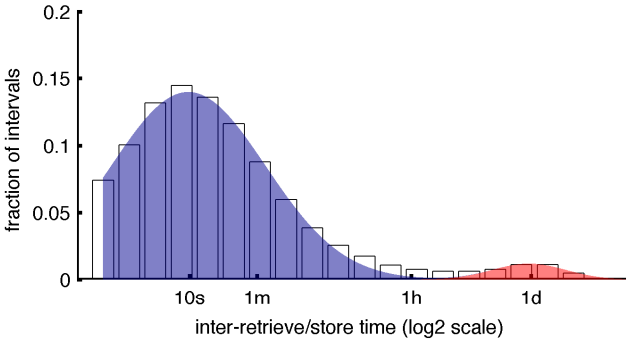


Figure 3: Histogram of time between sequential file operations of individual users (bars), which is fit with a mixture of Gaussians with two clusters: one for intra-session intervals and one for inter-session intervals.

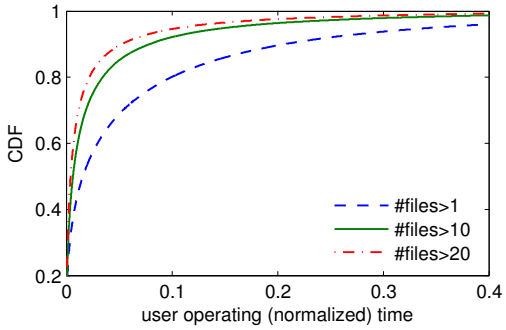


Figure 4: Cumulative distribution of user operating time, i.e. the time between the first file storage/retrieval operation and the last one in a session. The time is normalized by the session length.

By applying the session identification methodology, we obtain 2,377,124 sessions, which are further classified into 3 classes: *store-only* (i.e., containing only file storage requests), *retrieve-only* (i.e., containing only file retrieval requests) and *mixed* (i.e., containing both storage and retrieval requests). Surprisingly perhaps, we find that more than 68% of the sessions are store-only. On the other hand, only 2% of the sessions include both storage and retrieval processes, which strongly suggests that users tend to perform a single type of task within one single session. The dominance of store-only sessions implies that the mobile cloud storage service is *write-dominated*, which is quite different from traditional PC-based cloud storage services, which are read-dominated [12, 16].

3.1.2 Burstiness within sessions

To explore whether users perform all file operations at the beginning of sessions, we measure for each session the user operating time as the time between the first file operation request and the last one. Figure 4 depicts the distribution of user operating time, normalized by the session length as defined in Figure 2. Since in sessions containing only one file operation, file operations are always performed at the beginning of sessions, our analysis only considers those having more than 1 file operation, which are further divided into based on the number of file operations. Figure 4 shows that regardless of the number of file operations, for over 80% of the sessions, the normalized user operating time is below 0.1, indicating that users tend to perform all the file operations at the beginning of the sessions, then wait for the finish of the upload or download. Interestingly, the more files in a session, the higher the likelihood that all the file operation requests are sent early the session. For example, users issued all requests within 3% of the session length in the sessions with more than 20 file operations. This is likely caused by the mobile app that allows users to store or retrieve multiple files (e.g., batch backup of photos in mobile devices). The device then issues these file operation requests within a short time period.

The burstiness of store and retrieve activity within a session is a challenge for the load balance in the back-end servers [18]. In addition, since metadata servers are only used at the beginning of sessions, it is very important to decouple the metadata management and the data storage management (as opposed to involving metadata management during the whole session), in order to alleviate the load on metadata servers.

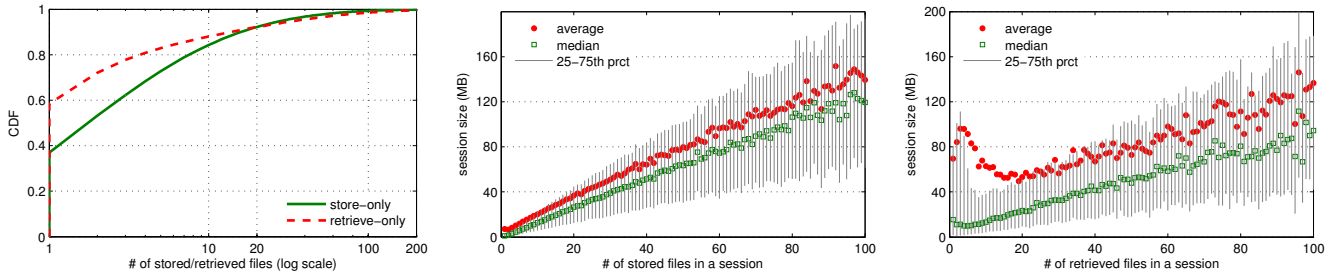
3.1.3 Session size

Next, we examine the session size, which is measured as the volume of data transferred in individual sessions during our observation period. We first show in Figure 5a the cumulative distribution (CDF) of the number of file operations in individual sessions[3]. We observe that users tend to retrieve or store very few files within a session, which is evidenced by the fact that, independent of session types, 40% of the sessions contain only one file operation. Nevertheless, around 10% of the sessions contain over 20 files, possibly corresponding to synchronization of multiple files in a directory to/from the cloud.

To analyze the data volume of individual sessions, we group sessions into bins, where each bin contains sessions that store or retrieve the same number of files. We then compute the average, median, 25th percentile and 75th percentile data volume over the sessions in each bin. Figure 5b and Figure 5c report the results for store-only and retrieve-only sessions respectively.

The linear relationship between data volume and number of stored files is visible for store-only sessions (Figure 5b). The linear coefficient is about 1.5 MB, which corresponds to the average file size. We conjecture that the prevalence of photo backup from mobile devices enabled by the mobile app results in this average file size. The cloud service provider confirmed that most of the uploads are personal photos taken by smart phones or tablets. This observation reveals a difference in usage scenarios between the mobile

[3]We do not show the results for mixed sessions since they are not significant.



(a) # of files in individual sessions (b) data volume: store-only sessions (c) data volume: retrieve-only sessions

Figure 5: Session size when varying the number of store/retrieve operations per session.

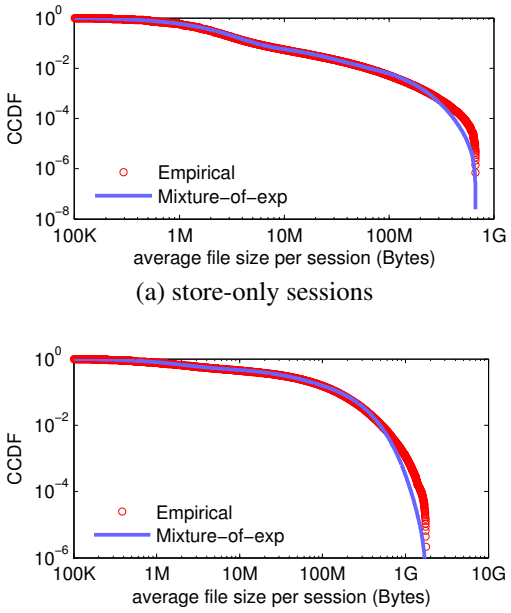
cloud storage service and the traditional ones, in which majority of files are very small ($<$ 100 KB) [21]. Besides, we note that for short sessions (e.g., fewer than 10 files stored), the variation of session data volume is small.

The data volume results for retrieve-only sessions (Figure 5c) show that they have very different characteristics. The average is even higher than the 75th percentile value for some bins, indicating some sessions in these bins transfer a huge amount of data. Moreover, the average session volume is surprisingly large when users retrieve only a few files within a session. For instance, the average volume is as large as about 70 MB in sessions retrieving only one file. This behavior could be the result of the use of cloud storage for file sharing. Some popular content (like videos or software packages) can be easily shared widely through URLs of the files. Downloading this content from a cloud platform, which is known to have good content delivery network coverage (like the one that we examine), might prove often faster than downloading from traditional web sites [19][23].

The difference in session size characteristics between storage and retrieval sessions suggest that they use the mobile storage service for different purposes. We further examine the usage scenarios later in this section. In addition, since the majority of the sessions include very few file operations, the possibility of bundling multiple files for upload will be low. Indeed, even for the sessions that contain dozens of file operations, the bundling will not significantly reduce communication overhead traffic (arising from TCP/HTTP connection setup and maintenance, metadata delivery, etc.), since the size of one single file is large enough to achieve efficient traffic usage [21]. Rather, due to the large file size, the cloud service uses multiple TCP connections to accelerate upload and download. However, cares should be taken when using multiple TCP connections on mobile devices because of power, memory and CPU constraints [9].

3.1.4 *Modeling the average file size*

To further investigate the attributes of stored and retrieved file, we compute the average file size for each session as the session data volume normalized by the number of files. Figure 6 plots the CCDF (complimentary CDF) of average file size for individual sessions. We observe a clearly heavy-tailed distribution for both types of sessions.



(a) store-only sessions

(b) retrieve-only sessions

Figure 6: Mixture exponential fit for average file size of individual sessions. Both axes are in logarithmic scale

Our previous analysis on session size indicates that users might upload or download files of several typical types, like photos and videos. To capture these different file types and account for the observed heavy-tailed distribution, we use the mixture-exponential distribution model [20, 27]. The PDF (probability distribution function) of a mixture-exponential distribution is

$$f(x) = \sum_{i=1}^{n} \alpha_i \frac{1}{\mu_i} e^{-\frac{1}{\mu_i}x}$$

where u_i (in MB) is the mean of the i-th exponential distribution, α_i represents the weight of the i-th component and $\sum_{i=1}^{n} \alpha_i = 1$. A prominent feature of this model is that each μ_i indicates a typical file size and α_i can be taken as the fraction of files with the corresponding size μ_i.

For each of the two session types, we iteratively determine the number of exponential distributions (*i.e.,* the value of n)

Table 2: Model parameters for average file size. μ_i is in MB.

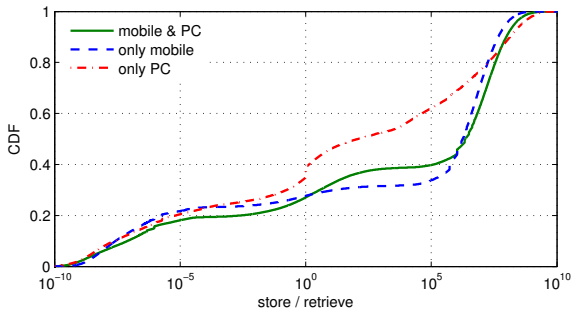
Sess. type	α_1	μ_1	α_2	μ_2	α_3	μ_3
store-only	0.91	1.5	0.07	13.1	0.02	77.4
retrieve-only	0.46	1.6	0.26	29.8	0.28	146.8

to be used in the mixture. Specifically, for each n, we use the EM algorithm [7] to find the maximum likelihood estimates of the parameters μ_i and α_i. We identify that $n = 3$ achieves a good match, as adding the fourth exponential distribution component leads to one of the α_i parameters being close to 0 (*i.e.,* < 0.001), indicating the negligible effect of this component. Table 2 lists the model parameters. The corresponding mixture-exponential distributions are also plotted in Figure 6, which visually shows that the developed models fit the empirical distributions quite well[4].

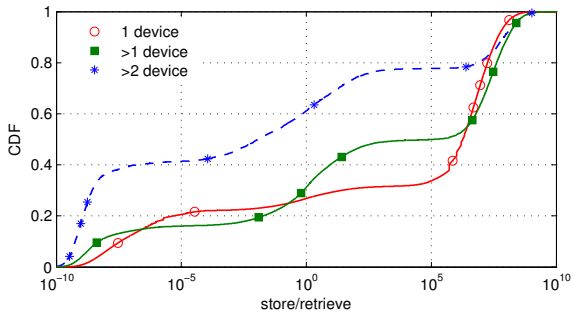
We speculate that the first component of the mixture distributions, captured by α_1 and μ_1, corresponds to the photo synchronization between mobile clients and the cloud storage service, as the average size indicated by μ_1 is about $1.5 \sim 1.6$ MB, corresponding to a typical 8Mp JPEG photo. The value of α_1 indicates that 91% of storage sessions were synchronizing this type of files, two times that of retrieve sessions. The second and third components for storage sessions, which account for less than 10% of cases in total, seem to be related to users uploading short and long videos recorded on their mobile devices. Retrieval sessions, on the other hand, tend to download larger files, which is evidenced by the fact that μ_2 and μ_3 are double those for storage sessions. In particular, 28% of the retrieved files are around 150 MB in size. They may be the result of downloading short video clips [19] based on URLs obtained from third-party sites, like social medias and online social networks.

Our combined observations on session and file size have several important implications. First, most of store-only sessions are backing up files of size around 1.5 MB, which is likely to be photos on mobile devices (as confirmed by the service provider). This implies that neither data compression nor delta encoding [11] can greatly improve system efficiency for the service. This is because, on the one hand, compression has a negligible effect on reducing photos size, and on the other hand, photos are immutable [24]. Second, a considerable fraction of retrievals (28%) download large files of about 150 MB. These downloads may take relatively long time, suggesting a need for resilience to possible failures, such as support for resuming a failed download, to avoid downloading from the beginning after failures that could be frequent for mobile network. Third, it would be necessary to monitor the popularity of downloads to verify whether there exist a locality of user interests, i.e. a handful of popular files dominate the downloads. If so, web cache proxies can reduce server workload and improve user perceived performance.

[4]We have also designed Chi-square goodness-of-fit tests for the fittings. Both fittings pass the test when considering the significant level of $P_0 = 5\%$.



(a) Mobile and PC



(b) Mobile Only

Figure 7: The ratio of stored data volume to retrieved data volume per user. (a) the impact of PC clients; (b) the impact of using multiple mobile devices.

3.2 Usage Patterns

We next look into different aspects of the storage service usage. Here, the analysis also uses the access logs from PC clients, so we can, for example, understands factors related to users who use multiple devices, either mobile or PC clients. Out of the 1,148,640 mobile users in our dataset, we identified 164,764 users that use both mobile devices and PCs.

3.2.1 Usage scenarios

The cloud storage service might be used for different purposes. For instance, some users might use it as a backup service, and thus mainly upload files, while other might use it as a content distribution platform (e.g., videos or software packages) and thus mainly download files on their devices. We use the ratio of stored data volume to retrieved data volume of each user to identify different usage scenarios. Figure 7a plots the cumulative distribution of the ratio, where we consider the differences between mobile users and PC users.

We can identify three usage patterns: *dominating retrieval* for those with a ratio below 10^{-5}, *dominating storage* for those with a ratio over 10^5, *mixed usage* of both storage and retrieval for those left. We observe that mobile users follow the dominating storage pattern more than PC users. In contrast, PC users have a higher likelihood to perform both storage and retrieval. Figure 7b further examines the usage patterns for users who use only one or more mobile devices. We

see that the number of mobile devices in use heavily impacts usage pattern. Specifically, there is a significant reduction in the storage dominating users when using multiple mobile devices. This observation can be the result of the frequent synchronization of personal data between multiple devices.

Inspired by [12], we classify users into four types: (i) occasional users with total data volume less than 1 MB; (ii) upload-only users with the stored/retrieved volume ratio lover 10^5; (iii) download-only users with the ratio less than 10^{-5}; and (iv) mixed users, who are not belonging to any of the other three types. Table 3 shows the percentage of users in each group, as well as the stored and retrieved data volume (relative to the total stored and retrieved volume during our observation period).

We see that over half of mobile users are classified as upload-only users and generated over 80% of the total storage volume. These users are predominantly interested in the cloud storage for backup of their personal data from their mobile devices. In contrast, only 7.2% of users fully exploit both storage and retrieval to synchronize files in both directions. This is further evidence that mobile users view the service more as a backup service. Download-only users, on the other hand, account for $15 \sim 17\%$. A typical scenario for this type of users is that users get URLs of file objects that they are interested in retrieving the content using the URLs directly from the cloud. In other words, they use the cloud storage service as a content distribution platform. Finally, occasional users contribute very little load. They may use the cloud storage service a few times but seldom come back. Another interesting observation from Table 3 is that the distribution of PC users across the four groups is much more evenly than mobile users. In particular, compared with mobile users, PC users are less likely to be classified into upload group, and a larger fraction of them fully exploit both storage and retrieval.

The identification of user patterns also shed light on target advertisements (ads). For example, since upload-only mobile users are likely to be more interested in taking photos or record video clips, the mobile terminals equipped with good cameras, and mobile apps with a good photo editor are potential candidates for in-app ads. Mobile cloud storage service providers can further improve the efficiency of advertisements by looking at the actual type of files.

3.2.2 User engagement

User engagement measures the possibility of users returning within a given period of time after their first visit. This metric reflects user dependence on the service and is critical for system optimization. We focus on the users that have at least one session in the first observation day, so that we have one week to observe their possible returns. In total, 233,225 users were active in the first day.

Figure 8 shows the statistics of user engagement, where we first stratify users based on whether PC clients are used, and then further divide those using only mobile devices into 3 groups based on the number of used devices. We observe a bimodal distribution for user engagement, where users either return the following day, or remain inactive for over a week

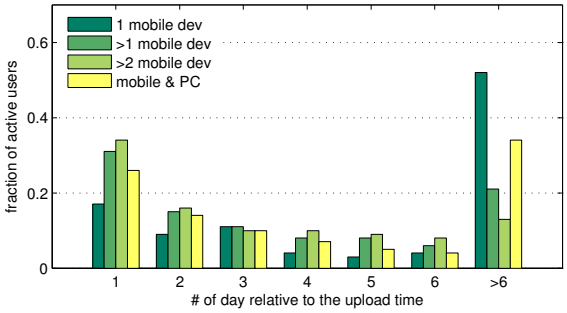


Figure 8: User engagement: fraction of users that are active on the first observation day and return back on the x-th day (relative to the first day).

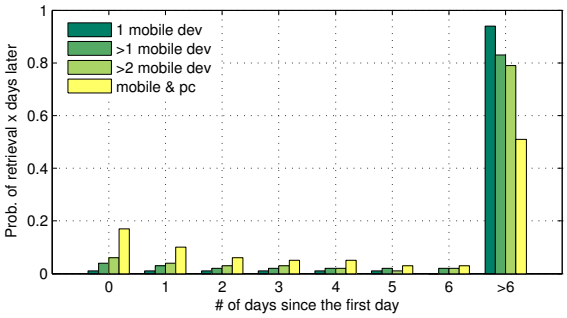


Figure 9: Fraction of users that uploaded files on the first day and have at least one retrieval session on the x-th day. Day 0 in the x-axis aligns to the first day.

during our observation period [6]. The impact of multiple devices is also notable. In particular, as many as half of the users that use only one mobile device remain inactive in the following week. This percentage drops greatly to less than 20% if multiple mobile devices are used. This is explained by the fact that users synchronize their data among multiple devices through the cloud storage service.

Since half of users are predominantly interested in uploading data, we then analyze the possibility of users returning back to retrieve the stored files that are uploaded in the first day. Since the file-related information is not available in our dataset, we instead examine the upper bound for this possibility by considering that any retrieval session after the storage session in the first day will retrieve the stored files.

Figure 9 depicts the upper bounds of the return possibility for four types of users. Surprisingly perhaps, independent of the number of mobile devices that were used by a user, over 80% of users that use only mobile devices will not retrieve any data in the week following the storage session. On the other hand, when both mobile devices and PC clients are used, the possibility of retrieving files in the following several days improves, especially on the same day of uploading (day 0 in the figure). This observation suggests that users are more likely to sync data uploaded by mobile devices from PCs than from another mobile device.

Table 3: Characteristics of four types of users: number of users, stored and retrieved data volume.

User Type	mobile only			mobile & PC			PC only		
	# users	store v.	retri. v.	# users	store v.	retri. v.	# users	store v.	retri. v.
upload-only	51.5%	86.6%	-	53.7%	81.3%	-	31.6%	74.8%	-
download-only	17.3%	-	84.5%	15.1%	-	66.5%	17.2%	-	75.5%
occasional	23.9%	-	-	13.2%	-	-	34.1%	-	-
mixed	7.2%	13.4%	15.5%	18.0%	18.7%	33.5%	19.1%	15.2%	14.5%

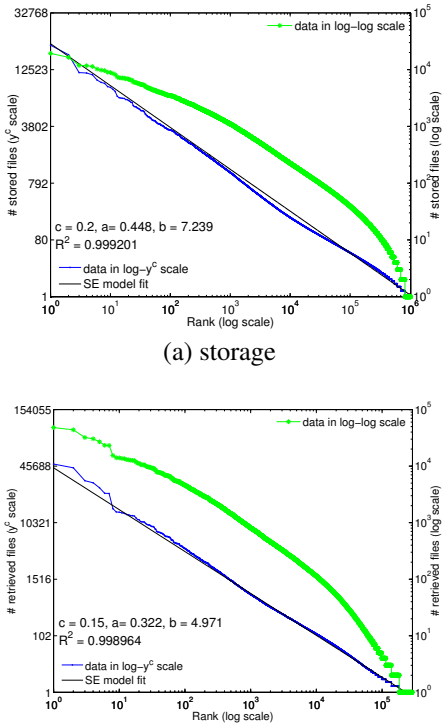
The low probability of downloading uploads within at least one week for vast majority of users indicates that most uploads could be deferred. The uploading deferment could cut down the cost greatly. For instance, the uploads during peak workload periods (e.g., 9pm to 11pm, see Figure 1) could be deferred to the following early mornings when the load is low. We thus posit a "smart" `auto backup` functionality for the implementation of uploading deferment, where users' files (like photos) are automatically backed up to the cloud (on users' permission) during the low-load and high connectivity periods (e.g. early morning with WiFi connectivity at home). Such functionality will implicitly limit manually uploads by users during peak hours. It also improves the mobile app's usability as users are relieved from manual file operations. Indeed, the deferment should be done carefully with the right disclosure for users, because there is a potential for hurting the users' QoE (Quality-of-Experience) if they are interested in reading some of the files soon after the uploading. Also, the effect of using multiple mobile devices and PC clients should be taken into account, since users may want to synchronize the data from another device very soon after uploading.

3.2.3 *User activity modeling*

Finally, we analyze the diversity of user activity, which is measured as the number of stored and retrieved files. We first examine whether user activity follows the power law distribution, which would be observed as a straight line in the rank distribution of user activity when plotted in a log-log scale. The power law distribution has been recognized in several online services [25]. However, we find from Figure 10 that the rank distribution of user activity deviates from a straight line in log-log plots (right y axis), indicating a non-power law distribution. Instead, user activity can be well modeled by a stretched exponential (SE) model. The CCDF of a SE distribution is given as follows

$$P(X \geq x) = e^{-(\frac{x}{x_0})^c}$$

where c is the stretched factor and x_0 is a constant parameter. Suppose we rank N users in a descending order of the number of stored (or retrieved) files, and the i-th ranked user stored (or retrieved) x_i files. Then, we have $P(X \geq x_i) = i/N$, meaning $\log(i/N) = -(\frac{x}{x_0})^c$ in the SE model. By substituting x_i for y_i, we have ${y_i}^c = -a \log i + b$, where $a = {x_0}^c$ and $b = {y_1}^c$. In other words, the log-y^c plot of ranked data that follows a SE distribution is a straight line. We obtained the parameters by maximum likelihood estimation following the method in [17].



(a) storage

(b) retrieval

Figure 10: Rank distribution for the number of stored (a) and retrieved (b) per user. The x-axis and right y-axis are in logarithmic scale, the left y-axis is in y^c scale (c is the stretch factor of SE model).

Figure 10 also plots the SE distributions with the estimated parameters; the left y-axis is in y^c scale. We observe that the SE models match the data quite well, as the log-y^c plots of ranked data (the blue lines) are close to straight lines. The coefficient of determination (*i.e.,* R^2), which measures the proportion of total variation of data explained by the model, further confirms the good fit. Comparing the SE model parameters between storage and retrieval, we observe a smaller stretched factor c for retrieval, meaning a more skewed distribution for retrieval activity. A possible reason is that the most active users might synchronize personal data from the cloud to multiple devices, and thus retrieve far more files than the low-ranked users.

The SE model implies that while the user activity is biased, the top ranked users are not as active as the power law predicts. This implies the influence of a small number of "core" users cannot dominate the system. As such, sys-

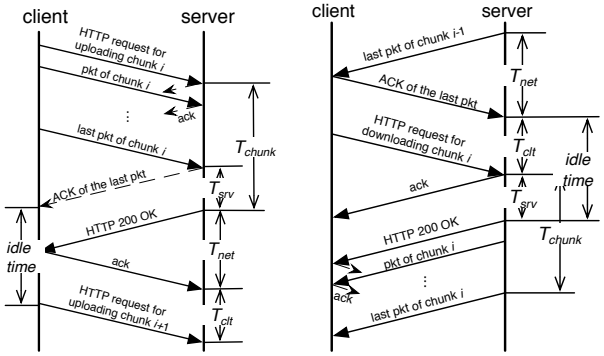
Table 4: Summary of major findings and implications in user behavior.

Major findings	Implications
Sessions: A two-component Gaussian mixture model captures the intra- and inter-session intervals. And over 68% of sessions are used only for storing files.	Sessions, which can be identified using the interval threshold derived from the model, are write-dominated.
Activity burstiness: Mobile users store and retrieve files in a bursty way.	It is necessary to decouple the metadata management and the data storage management in mobile cloud storage.
Session size: The majority of the sessions include very few file operations.	The possibility of bundling multiple files for transmission is very low.
File attribute: Over 90% of storage sessions are used to store file objects of 1.5 MB, which are likely to be photos on mobile devices.	Data compression and delta encoding are not necessary in mobile cloud storage services.
Usage pattern: Over half of the mobile users are predominantly interested in uploading objects and seldom retrieve data. In contrast, PC-based users are far more likely to fully exploit both storage and retrieval processes.	Mobile users are likely to use the service for backing up personal data. In other words, mobile users use the cloud storage service quite differently than PC-based users.
User engagement: Independent of the number of mobile devices in use, about 80% of the mobile users will not return in the following week to retrieve their uploads.	Uploads can be deferred to avoid the peak load period, and the cold/warm storage solution (e.g. f4 [26]) can cut the cost down significantly.
User activity model: The diversity of user activity is captured by a stretched exponential distribution, rather than a power law distribution.	System optimizations (like distributed caching, data prefetching [16]) that aim to cover "core" users should consider more users than that computed by a power law model.

tem optimizations (like distributed caching, data prefetching [16]) that aim to cover "core" users need to account for more users. Besides, mobile cloud storage system designers as well as researchers can leverage the SE models developed here for workload generation.

3.3 Summary and Implications

Table 5 summarizes the findings and implications of this section, in order to shed further light on the service optimizations. The findings consistently suggest that the mobile cloud storage service is upload-dominated. It seems that mobile users are more likely to use the mobile cloud storage as a backup service than PC-based users. Mobile cloud storage service providers and system designers can take the implications for efficiency optimization, cost reduction and revenue improvement.



(a) storage (b) retrieval

Figure 11: Timeline of storage and retrieval within a TCP flow.

4. DATA TRANSMISSION PERFORMANCE ANALYSIS

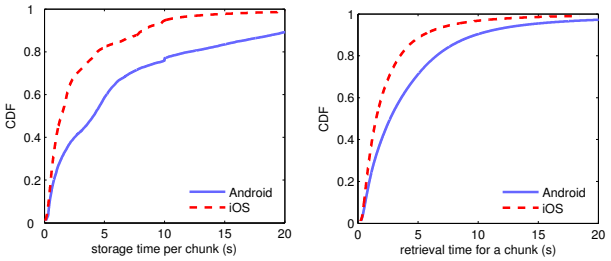
In this section, we examine the data transmission performance of the mobile cloud storage service, with an emphasis on the factors that limit the performance. Both HTTP access logs and the packet-level traces collected from the storage front-end servers are used. To eliminate the effect of HTTP proxies on the analysis, we filtered out those requests that were proxied by at least one proxy.

To provide a proper context for the analysis, Figure 11 depicts the timeline of uploading and downloading chunks, with important metrics marked. Each access log in our dataset contains the total request processing time by front-end server (T_{chunk}) as well as the upstream processing time (T_{srv}). In particular, T_{chunk} measures the duration between the first bytes received by front-end server and the last bytes sent to mobile client, while T_{srv} refers to the time spent in storing/preparing the requested content by upstream storage servers, i.e., the servers that physically host the data. The other two metrics T_{net} and T_{clt} can only be extracted from the packet-level traces, where T_{net} measures RTT (Round Trip Time), while T_{clt} is the time required by client to prepare the next chunk in the case of uploading, or to process the data of the latest downloaded chunk in the case of downloading.

Our main observations in this section include: (1) small receive window size advertised by servers limits the upload-

ing performance; (2) mobile device type has a notable impact on performance due to the difference in idle time between chunk transmissions; (3) client side processing time is the major contributing factor to the long idle time. Up to 60% of idle intervals in Android storage flows trigger the restart of TCP slow-start, which results in inferior performance.

4.1 Chunk-level Performance



(a) uploading (b) downloading

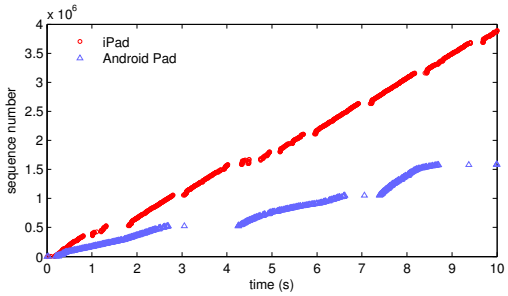
Figure 12: CDF for the time required to upload/download a chunk. Note the difference of x-axis scale in two subfigures.

We first examine the upload and download time of a chunk perceived by users, which is approximated as $t_{tran} = T_{chunk} - T_{srv}$ (see Figure 11). Figure 12 plots the distribution of transmit time for individual chunks recorded in the HTTP request logs. We surprisingly observe a significant longer time required by Android devices, especially for uploading. For instance, the median time for uploading is 1.6s for iOS devices, but as long as 4.1s for Android devices.

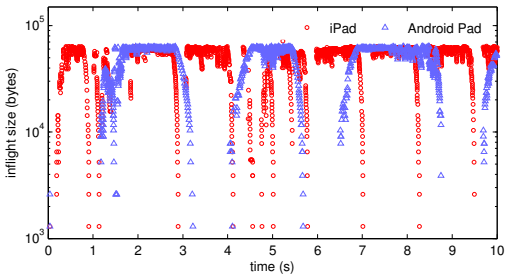
Since servers do not distinguish between device types, we conjecture that the throughput gap between Android devices and iOS devices come from the client side behavior. To this end, we conducted a series of active measurements. In particular, we connected a Samsung Pad (Android 4.1.2) and an iPad Air2 (iOS 8.4.1) with the mobile app installed to the Internet through a laptop, which acted as an AP (Access Point) sitting very close to our experimental devices. The laptop itself was connected to the Internet through WiFi. Files of the same size were uploaded or downloaded at the same time from two devices. We experimented with three typical file sizes: 2 MB, 10 MB and 80 MB. Note that we did not control bandwidth in the experiments. Packet level traces in `pcap` format were dumped from the laptop for analysis. We find that both types of devices connected to the same front-end server.

We observe for both storage and retrieval flows, Android clients take a longer time between two consecutive chunks than iOS clients. The long idle time between chunks significantly degrades the chunk transmission performance. To illustrate this, we compare in Figure 13 the sequence number over time and the inflight size (the number of bytes in flight) of an Android and iOS storage flow[5]. The inflight

[5]To better interpret the results, we show only the first 10 seconds of the flows.



(a) Sequence number



(b) In-flight size

Figure 13: Sequence number (a) and in-flight size (b) of a storage TCP flow observed at the client side.

size is an accurate estimation of the sending window at the TCP sender (which is client in uploading) [30], which determines the TCP throughput. On each ACK from the server, we compute the inflight size as the gap between the sequence number in the last packet sent by the client and the ACKed sequence number by the server.

We make two notable observations from Figure 13a. First, the iPad experienced higher throughput than the Android Pad. Second, the idle time between chunks in the Android flow can be over 1 second, much larger than that in the iPad flow. The large idle time of Android Pad indeed significantly degrades the performance as shown in Figure 13b, which shows the variation of inflight size over time, where the y-axis is in logarithmical scale. The long idle time between chunks in the Android flow is notable. Note that at the end of each chunk, the inflight size drops because the client has no data to send before the application-level acknowledgement (i.e., HTTP 200 OK) from the server for the current chunk.

Except the first chunk, the iPad begins the upload of each chunk with a sending window size close to 64 KB, the same as that at the end of the previous chunk. However, in the Android flow, each chunk begins transmission with a small sending window and takes some time to reach 64 KB in the previous chunk. This difference results from TCP behavior in the case of long idle time. TCP recommends resetting the congestion window to its initial value and begin with slow start if TCP is idle (i.e., no data has been sent) for an interval exceeding an RTO (Retransmission Timeout) [1]. As we will see later, 60% of the idle intervals between two chunks in Android storage flows exceed an RTO, while this percent-

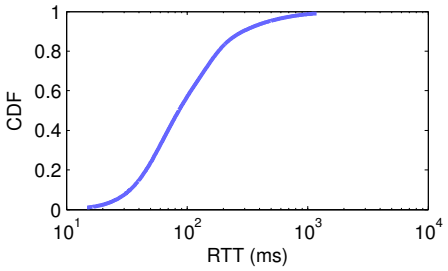


Figure 14: CDF of the RTT measured in the transmission of chunk. Note the x-axis is in logarithmical scale.

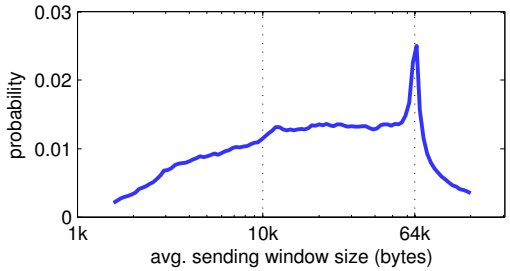


Figure 15: Probability distribution of the estimated average sending window size at client for storage flows.

age is only 18% for iOS flows. Given that the median RTT is around 100ms (see Figure 14), these Android flows will require as much as 0.5s (i.e., 5 RTTs) of extra time to reach a window size of 64 KB.

Another interesting observation from Figure 13b is that the inflight size is limited at about 64 KB. After examining the traces, we figured out this limitation is caused by the receive window size advertised by the TCP receiver (which is server). Indeed, in TCP without the window scaling option, the receive window size is 65,536 bytes at most [3]. In the cloud storage service that we examine, servers do not allow the window scaling option. To further verify this performance bottleneck, we estimate the average sending window size ($\overline{swnd}$) of upload flows using our dataset of HTTP access logs. The average performance of a TCP flow can be approximated as $\overline{swnd}/\overline{RTT}$, where $\overline{RTT}$ is the average RTT of the flow. As such, we have $\overline{swnd}/\overline{RTT} = req_{size}/t_{tran}$, where req_{size} is the volume of transmitted data of the request, and $t_{tran} = T_{chunk} - T_{srv}$. That said, $\overline{swnd} = req_{size} * \overline{RTT}/t_{tran}$.

Figure 15 plots the probability distribution of $\overline{swnd}$ that is estimated using individual access logs. The concentration around 64 KB is notable. This observation confirms that the sending window size is limited by the advertised receive window size of servers that disable the window scaling option. We have also examined the receive window size effect for the retrieval flows and found that mobile clients, which are TCP receivers when retrieving, enable the window scaling option. In fact, the advertised receive window by the Samsung Pad is as large as 4 MB, while it is 2 MB for the iPad. Such a huge receive window, however, might not be fully utilized and would result in waste of resources [9].

4.2 Dissecting Idle Time between Chunks

Next, we use the packet-level traces collected at front-end servers to make an in-depth analysis of the idle time between transmissions of two consecutive chunks. Here, the idle time refers to the TCP idle time, which is the time interval in which the TCP sender has not sent any data. As shown in Figure 11, the idle time at the TCP sender for both storage and retrieval is the sum of server processing time T_{srv} and client processing time T_{clt}. While Drago et al. [12] have shown that the sequential acknowledgment impairs *overall TCP flow throughput* due to the waiting for application-layer acknowledgments, we further reveal that the idle time between chunk transmissions can even heavily hurt the *throughput of individual chunks*.

We first show the distribution of T_{srv} and T_{clt} for storage flows in Figure 16a and retrieval flows in Figure 16b. Regardless of device type (Android or iOS) and flow type (storage or retrieval), the processing time at the server side is around 100ms, reconfirming that servers do not distinguish between device types. However, the processing time at the client side of Android devices differs significantly from that of iOS devices. In particular, Android devices spend on average 90ms more time than iOS devices in preparing data for the next uploading chunk. While the median processing time at the client side for retrieval flows of two types of devices is similar, notably, the 90th percentile time for Android devices is as high as 1s, one order of magnitude larger than that for iOS devices. We conclude that client side processing time (T_{clt}) is the major contributing factor to the long idle time in Android flows.

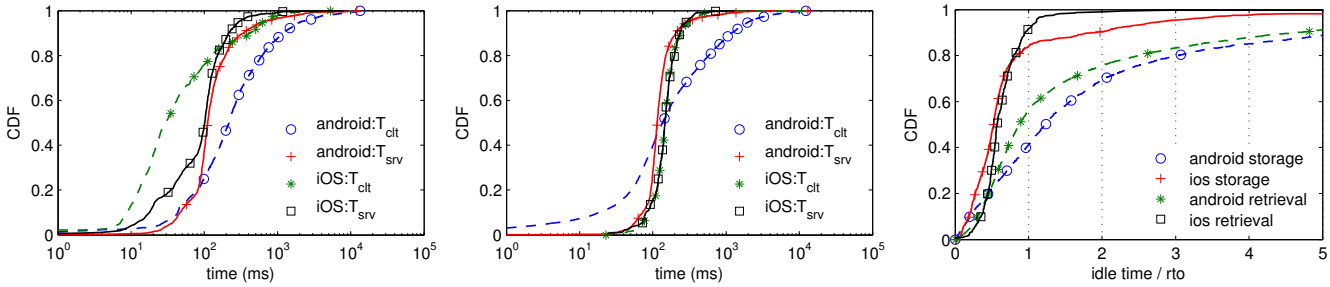
The long idle time between chunks of Android flows would trigger the restart of TCP slow-start [1]. To evaluate this, we examine the ratio of idle time ($T_{srv} + T_{clt}$) to the estimated RTO ($\widehat{RTO}$). In TCP implementation [2], RTO is computed as SRTT + $max(200ms, 4$ RTTVAR$)$, where SRTT is close to RTT, and RTTVAR is approximately $RTT/2$, *i.e.,*

$$\widehat{RTO} \approx RTT + max(200ms, 2RTT)$$

Figure 16c depicts the distribution for the ratio of idle time to RTO. It is notable that Android flows experience a much higher probability of restarts of TCP slow-start. In particular, about 60% of Android uploading chunks that are preceded by other chunks will start transmission with TCP slow-start, while this percentage is only 18% for iOS flows. We observe a similar gap for retrieval flows. This huge gap accounts for the performance difference between Android and iOS flows shown in Figure 12.

4.3 Summary and Implications

We have observed that the small receive window size is a factor that limits the performance of storage flows for both iOS and Android devices. This will greatly hurt QoE of the mobile cloud storage service. A straightforward solution is to enable the window scaling option at the server side. However, service providers should be aware of the cost of this option when serving million of concurrent flows. First, higher



(a) storage (b) retrieval (c) idle time / RTO

Figure 16: Dissecting the idle time between two consecutive chunks: (a) storage flows, (b) retrieval flows, (c) the ratio of idle time to RTO.

bandwidth is required to support the improved throughput of individual flows. Second, if the operating system kernel at the server side preallocates the memory for sockets, the large receive window size will lead to increased memory requirements and a possible waste of resources in the case that throughput is limited by network or client side factors, rather than the receive window advertised by servers.

The effect of long idle time between chunks on TCP behavior of restarting slow-start may be mitigated by simply disabling the implementation of slow start after idle (SSAI). However, without SSAI, the connection is likely allowed to send out a large burst after the idle period. In this case, packet loss may happen, especially for the packets at the tail of the burst. Once packets at the tail of the burst are lost, expensive timeout retransmission may be need for loss recovery [13], which yields low performance. Rather, optimizations that aim at improving TCP start-up performance might be useful [28][29]. For example, some packets can be paced out at a certain rate until the ACK clock can be restarted [28]. Moreover, since client side processing time dominates the idle time (see Figure 16), system operators should investigate the causes of long processing time on Android clients and then shorten it.

On the other hand, the effect can also be mitigated by reducing the number of intervals between chunk transmissions. To this end, a larger chunk size can be used. Our analysis has revealed that users tend to synchronize files of size over 1.5 MB, so increasing the chunk size from 512 KB to 1.5∼2 MB is indeed reasonable. In addition, batch commands that allow several chunks to be transmitted in a single request will also reduce the number of intervals in flows [12].

5. DISCUSSION

Threats to validity. Although our dataset consists of 349 million logs from over 1 million users, it is from only one service provider. While our findings clearly reveal the disparity between mobile users and PC-based users in using the cloud storage services, care should be given when generalizing our findings to other mobile cloud storage services. Another limitation of our study is that our observation period is only one week. For example, we cannot distinguish between lack of downloads and infrequent downloads when identifying individual users' behavior. To reduce these threats, we have made our dataset publicly available for the community to further valid our findings.

Usage of mobile cloud storage. Our analysis raises an interesting question: Is the mobile cloud storage a backup service, rather than a file hosting service? The findings in this paper consistently show that the examined service is upload-dominated, and users access their uploads infrequently. However, since the dataset does not contain any file or chunk identifications, this work has not fully answered this question. We plan to further explore this question.

6. RELATED WORK

The usage pattern of Dropbox was first examined in [12], where the performance degradation caused by the sequential acknowledgment is also examined. The authors further extended their analysis to 5 cloud storage services in [11]. Mathematical models for Dropbox sessions were developed in [14]. Bocchi *et al.* [8] compared three services (i.e., Dropbox, Google Drive and OneDrive) by passively observing traffic, showing that users of each service exhibit distinct behaviors. Authors in [15], on the other hand, actively measured three cloud storage platforms (i.e., DropBox, Box and SugarSync) with a focus on the file mean transfer speed. Liu *et al.* [24] examined the access patterns in a campus-wide cloud storage system, and found most of files are rarely accessed. These studies largely focus on traditional PC-based cloud storage platforms, so the observations might not be applicable to mobile ones as we have shown throughout this paper. Our work also extends these studies by examining all users in a large-scale service, rather than users in small regions (e.g. within a university campus) as they analyzed.

A recent study on Ubuntu One (U1) [16] considered all users of the service by examining logs from *metadata* servers, with a focus on the back-end activities and performance. In contrast, our work examines requests logs from *data storage front-end servers*, and thus has a unique view of data transmission behavior and performance. Besides, due to the mobile usage, the service that this paper examines shows

distinct usage patterns. For instance, the examined service is write-dominated, while U1 is read-dominated.

A particular concern in cloud storage services is the unnecessary traffic generated by synchronization caused by file editing, where any change at either client side or server side will be automatically synchronized to the other side. Li *et al.* [22] studied the traffic overuse problem in Dropbox, i.e., session maintenance traffic far exceeds the useful update traffic. The authors extended their analysis to 6 popular services to identify the general factors that may affect the data synchronization traffic [21]. In the same vein, QuickSync, which consists of network-aware chunker, redundancy eliminator and batched syncer, was proposed in [10] to enable efficient synchronization for mobile cloud storage. As we have found, mobile users tend to consider cloud storage services as backup services and most of files seem to be immutable, the traffic overuse problem caused by frequent file editing might be negligible.

7. CONCLUSION

This paper examines data of HTTP requests from mobile devices in a large-scale cloud storage service, to study the system's artifacts and data transmission performance. Our results suggest a backup-dominated usage pattern for mobile users. This is evidenced by a number of observations, including a write-dominated behavior at both the session and user level, and infrequent retrieval of uploads. As for data transmission performance, the small receive window advertised by servers is a potential factor that limits the backup performance. Perhaps more importantly, the long idle time between chunk transmissions, which is much more significant in Android flows, triggers the restart of TCP slow-start, and thus greatly hurt performance. We also discussed the implications of these findings on system design, application development and transmission optimization. The implications provide guidance to mobile cloud providers to cut down the cost, increase indirect revenue and improve performance.

Acknowledgement

The authors would like to thank our shepherd Krishna Gummadi and anonymous reviewers for their feedback. This work was supported in part by National Basic Research Program of China with Grant 2012CB315801, by High-Tech Research and Development Program of China ("863-China Cloud" Major Program) with Grant 2015AA01A201, by National Natural Science Foundation of China with Grants 61572475 and 61272473, by CCF-Tencent Open Fund with Grants IAGR20150101 and IAGR20160101.

8. REFERENCES

[1] Rfc5681: Tcp congestion control. https://www.ietf.org/rfc/rfc5681.txt, 2009.

[2] Rfc6298: Computing tcp's retransmission timer. https://www.ietf.org/rfc/rfc6298.txt, 2011.

[3] Rfc6298: Tcp extensions for high performance. https://tools.ietf.org/html/rfc7323, 2014.

[4] Dropbox highlights. https://www.dropbox.com/news, 2016.

[5] World personal cloud market, 2014 -2020. https://www.alliedmarketresearch.com/personal-cloud-market, 2016.

[6] A. Balasubramanian, N. Balasubramanian, S. J. Huston, D. Metzler, and D. J. Wetherall. Findall: A local search engine for mobile phones. In *Proceedings of the ACM CoNEXT*, 2012.

[7] T. Benaglia, D. Chauveau, D. R. Hunter, and D. S. Young. mixtools: An r package for analyzing mixture models. *Journal of Statistical Software*, 32(6), 2009.

[8] E. Bocchi, I. Drago, and M. Mellia. Personal cloud storage: Usage, performance and impact of terminals. In *Proceedings of the IEEE CloudNet*, 2015.

[9] X. Chen, R. Jin, K. Suh, B. Wang, and W. Wei. Network performance of smart mobile handhelds in a university campus wifi network. In *Proceedings of the ACM IMC*, 2012.

[10] Y. Cui, Z. Lai, X. Wang, N. Dai, and C. Miao. Quicksync: Improving synchronization efficiency for mobile cloud storage services. In *Proceedings of the ACM MobiCom*, 2015.

[11] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras. Benchmarking personal cloud storage. In *Proceedings of the ACM IMC*, 2013.

[12] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras. Inside dropbox: Understanding personal cloud storage services. In *Proceedings of the ACM IMC*, 2012.

[13] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan. Reducing web latency: The virtue of gentle aggression. In *Proceedings of the ACM SIGCOMM*, 2013.

[14] G. Goncalves, I. Drago, A. Couto da Silva, A. Borges Vieira, and J. Almeida. Modeling the dropbox client behavior. In *Proceedings of IEEE ICC*, 2014.

[15] R. Gracia-Tinedo, M. Sanchez Artigas, A. Moreno-Martinez, C. Cotes, and P. Garcia Lopez. Actively measuring personal cloud storage. In *Proceedings of IEEE CLOUD*, 2013.

[16] R. Gracia-Tinedo, Y. Tian, J. Sampé, H. Harkous, J. Lenton, P. García-López, M. Sánchez-Artigas, and M. Vukolic. Dissecting ubuntuone: Autopsy of a global-scale personal cloud back-end. In *Proceedings of the ACM IMC*, 2015.

[17] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao. Analyzing patterns of user content generation in online social networks. In *Proceedings of the ACM KDD*, 2009.

[18] A. Halfaker, O. Keyes, D. Kluver, J. Thebault-Spieker, T. Nguyen, K. Shores, A. Uduwage, and M. Warncke-Wang. User session identification based on strong regularities in inter-activity time. In

Proceedings of the International Conference on WWW, 2015.

[19] Y. Huang, Z. Li, G. Liu, and Y. Dai. Cloud download: Using cloud utilities to achieve high-quality content distribution for unpopular videos. In *Proceedings of the ACM MM*, 2011.

[20] N. P. Jewell. Mixtures of exponential distributions. *Ann. Statist.*, 10(2), 06 1982.

[21] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, and Z.-L. Zhang. Towards network-level efficiency for cloud storage services. In *Proceedings of the ACM IMC*, 2014.

[22] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai. Efficient batched synchronization in dropbox-like cloud storage services. In *Proceedings of the ACM/IFIP/USENIX Middleware*, 2013.

[23] Z. Li, C. Wilson, T. Xu, Y. Liu, Z. Lu, and Y. Wang. Offline downloading in china: A comparative study. In *Proceedings of the ACM IMC*, 2015.

[24] S. Liu, X. Huang, H. Fu, and G. Yang. Understanding data characteristics and access patterns in a cloud storage system. In *Proceedings of the IEEE/ACM CCGrid*, 2013.

[25] L. Muchnik, S. Pei, L. C. Parra, S. D. S. Reis, J. Andrade Jr, S. Havlin, and H. A. Makse. Origins of power-law degree distribution in the heterogeneity of human activity in social networks. *Scientific Reports*, 3, 2013.

[26] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang, and S. Kumar. f4: Facebook's warm blob storage system. In *Proceedings of the OSDI*, 2014.

[27] T. Qiu, Z. Ge, S. Lee, J. Wang, J. Xu, and Q. Zhao. Modeling user activities in a large iptv system. In *Proceedings of the ACM IMC*, 2009.

[28] V. Visweswaraiah and J. Heidemann. Improving restart of idle TCP connections. Technical report, University of Southern California, Computer Science Department, 1997.

[29] Y. Zhang, L. Qiu, and S. Keshav. Optimizing tcp start-up performance. Technical report, Cornell University, Computer Science, 1999.

[30] J. Zhou, Q. Wu, Z. Li, S. Uhlig, P. Steenkiste, J. Chen, and G. Xie. Demystifying and mitigating tcp stalls at the server side. In *Proceedings of the ACM CoNext*, 2015.