The Time Overhead Inside the Data Sync Procedure of Personal Cloud Storage Service

Haihua Zhong

School of Software, Tsinghua University, Beijing 100084, China

Zhonghh09@gmail.com

Keywords: Time Overhead, Data Sync, Network, Measurement, Dropbox

Abstract. In recent years, Personal Cloud Storage(PCS) Service like Dropbox has achieved a great business success and has attracted a large interest from academia. However, considering the security of users' data, the PCS applications communicate with their servers by HTTPS protocol, which makes it hard for researchers to understanding how data sync works at the level of HTTP connection. As a result, the time overhead inside the procedure is unclear from the client view. Different from the previous work, by dynamic injection to the client and active measurement, we verify what's the role of each HTTP connection in the network communication procedure when the PCS applications synchronize the data to the server. As a result, we reveal the time overhead inside the procedure, and characterize the time overhead generated from both the processing and the network. Interestingly, the time overhead is different between the data upload and download in a real network with proxy servers. We believe that the findings could be an enlightenment for developing a more cost-effective PCS application.

1. Introduction

Personal cloud storage services like Dropbox, Google Drive and OneDrive has achieved a great success since they appeared in the Market. Dropbox, one of the most important players, owned more than 500 million users around the world and 1.2 billion files are uploaded to it everyday. (https://www.dropbox.com/news/company-info) According to the research, at the same network the traffic generated by Dropbox has occupied about 1/3 of that by YouTube, which is the most famous streaming service, and is still growing fast [1]. The other players also achieve similar success. People can sync and share photos, documents and files of in other data format in the folder between different devices from anywhere silently, quickly and safely through the PCS services.

The popularity of PCS applications has attracted a large interest from academia. Data synchronization is the most important operation in the PCS service, which uploads data to the cloud storage server immediately after the data updates and downloads the data updated by other users. By passive measurement using a Squid proxy server, researchers have revealed many characters of the PCS applications and found the common network communication procedure of the data sync. However, as it detects the network flows at a middle node of the connections generated by PCS client, it cannot give an exact description of how long it takes for each step from the moment of the file modification to the moment of the changed data synced on the other client. The time overhead of these steps, especially those steps involving network connections, is an important parameter of improving the performance of PCS services.

In this paper, we study the HTTPS connections by dynamically injecting some code to the client and jacking SSL communication while Dropbox is syncing data. As a result, we verify the network communication procedure of syncing data with different size and how they do it by analyzing the key parameters of each request. At the same time, we find out how long it takes for each HTTP request to complete its task and give a reasonable explanation for the phenomenon. The whole process contains two part, the *processing stage* and the *network stage*. Correspondingly, the period can be tagged *processing time* and *network time*. We summary the key findings as follows:

- The *processing time* is constant about 10~11s and the *network time* is increased with the size of synced data but not proportionally.

- The *startup* time (i.e. the time between the file finishing changing and the first connection initiated) of file with small size is longer than that with big size.
- Besides the *startup* time, the time overhead before data flow and the time overhead by the server dominate the the processing time, which can reach more than 80%.
- In total, the client spends more time on data uploading than data downloading in a real scene of communicating by a proxy server with limited bandwidth. What's more, the larger the file size is, the larger the time difference is.
- No matter how much the size of file uploaded is, the time of control network flows is close.

We believe that these finds could be an enlightenment for improving the performance of Dropbox-like PCS service.

Taking into account the market share, we mainly pay attention to Dropbox, but the findings can be a good reference for other similar services. In this paper, we will talk about the related work in Sec. 2. And after the introduction of background and methodology in Sec. 3, we give an overview of the time overhead in Sec. 4. Sec. 5 and Sec. 6 show the time overhead of processing and network correspondingly. Finally, we give a summary of this paper.

2. Related Work

As descripted above, this paper focus on the time overhead inside the data sync procedure of PCS service. *Drago* et al give a detailed measurement of Dropbox by setting up a Squid proxy server and using the SSL-bump, which shows the network characters of Dropbox [1]. *Drago* et al also give a benchmark comparison with five popular PCS services [2]. Goncalves construct a model of the workload generated by Dropbox client [8]. Because they detect the network flows at the middle node in the network connection, they do not talk about the problem of time overhead. Different from the method they use, we detect the network flows by injecting code to the client and with the client running in the environment of *DynamoRIO*, which is a runtime code manipulation system [9]. This method makes it precise to record the time of network activities.

Li Z et al study the bottleneck of Dropbox when facing the large small files update [4]. In [5] Li Z et al talk about the TUE (traffic usage efficiency) of PCS services. And in [6] Li Z compare the pros and the cons of PCS services and provide some chooses for designing a better service. Again they do not focus on the time overhead inside the data sync procedure.

Kholia et al provide a technique called code injection for intercepting SSL data [7]. However, they do not analyze anything about the SSL data. Besides, they provide a method to reverse the Dropbox application. In [8] Cui Y et al explained the sync inefficiency problem by using this method. In this paper, we will use the technique to get the accurate time of the network connection happening.

Besides, Zhang Y et al conduct a good study on the relationship between the PCS and the file system [10, 11]. And [12] descripts the architecture of the Dropbox storage server.

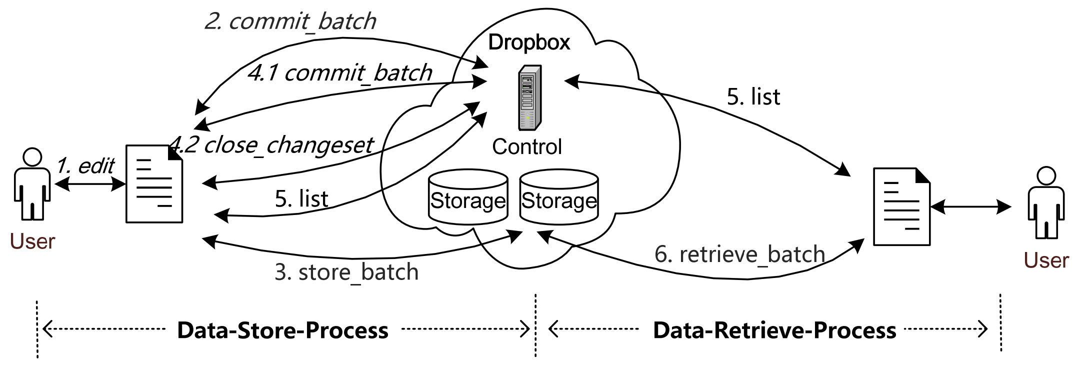


Fig. 1 the basic procedure of data sync

3. Background and Methodology

3.1 Dropbox Background

Dropbox is the most popular PCS at this time. The Dropbox client monitors the sync folder created by itself, and immediately syncs the data to the server in America. The server contains two components， the *control* server and the *storage* server. The former manages the meta data of the files and users; the latter stores the content of files. Formerly the storage server is constructed on the Amazon S3, but now Dropbox migrates it out and builds the cluster itself. The centralized architecture of Dropbox results in slow Round Trip Time (RTT) for the client far from the server. In our experiment, the RTT is about 230ms and the min RTT is about 157ms.

To reduce the network traffic at most, Dropbox divides the big file into 4MB chunks to realize incremental sync and bundles the small files into a big data chunk to reduce the creation and destruction of network connections. Besides, Dropbox maintains two connection pools, each with 4 HTTP connections. One is for uploading, the other is for downloading.

We verify the data sync procedure of Dropbox. Fig. 1 shows the basic complete procedure of data sync from the Dropbox client side. In the *data-store* process, after the user editing the file, the client sets up a *commit_batch* connection to report the meta data to the Dropbox control server and receives response by the *need block*s. Then the client uploads the data blocks to the storage server. The process ends with the second *commit_batch* connection to check the end state of the data uploading and a *close_changeset* connection to close the transaction. A *list* connection is set up soon after the response of a *notify* connection is received which is happened after the second *commit_batch* connection. This connection is used to fetch up the newest state of the server to decide whether set up a *data-retrieve* process or not. In the *data-retrieve* process, a *list* connection is set up almost at the same time after the previous one. And then, the data is retrieved by the *retrieve_batch* connection.

3.2 Code Injection

As mentioned above, we use the technique called code injection for intercepting SSL data from the client side. We modify the code provided by the author to record the immediate time of the network connections. The code prints the time before the data written into the SSL connection and after the data read from the SSL connection. In this paper, we use the DynamoRIO to construct the *.so* file, inject it to the Dropbox application with running the Dropbox application in the environment of DynamoRIO.

3.3 Controlled Measurement

We generate randomly 6 groups of files with the size of 0KB, 1KB, 10KB, 100KB, 1MB and 5MB. They are created in the Dropbox folder until about 30s ~ 60s after the previous one has finished syncing. We conduct the measurement on two desktops installed the 64 bit Ubuntu 14.04 with a dual-core Intel processor @3.2 GHz, 4 GB of RAM in Being. The bandwidth is 100 Mbps, which will not be the bottleneck of the sync procedure. As Dropbox cannot be accessed in our country, a proxy server located in America with a limited bandwidth of 1Mbps is used, which is a common typical real scene. Each group will be tested 10 times.

We also conduct the test with file modification with different size at different location of file, as the sync procedure can be treated as the file creation with corresponding size, we will not talk about the result specially.

4. The Time Overhead Overview

As we can see from the Fig. 2, The whole procedure can be divided into the processing part and the network part (those dark area). The processing time contains several parts separated by network connections. It keeps constant between the sync procedures of different file size, which is about 10s. The time spent on the network connection increases from about 1s for 0KB file to 70s for 5MB file. We can simply find that the increment is not Proportional to the file size. There are two reasons. When the file size is small, the RTT is the main part of the connection. While the file size is large, it is because Dropbox keeps 4 HTTP connections in the connection pool which can sync data simultaneously. However, the effort of Dropbox can be canceled and the network time seems a little high for the bandwidth of the proxy is only 1Mbps which restricts the data transferring much. However, this is actually common for those countries in which Dropbox cannot be accessed normally.

5. The Time Overhead of Processing

5.1 Startup

The startup time is referred to be the time between the file finishing setting the modification time and the first network connection started. This metric is important because it reflects the time overhead for Dropbox preparing for the data sync. By Fig. 2, We find that the startup time of 0KB file and 1KB file is longer than that of other larger file. As mentioned above, the reason could be that Dropbox bundles small file to a bigger data chunk to sync, which makes it wait another file for some time. This is reserved for the traffic saving, especially for those hundreds of small files added to the sync folder at the same time, and it is a good balance between the traffic overhead and the sync speed. Compared to the startup time, the time cost of the end stage is shorter and more stable. Obviously, this may hurt the user experience when they need a real-time service.

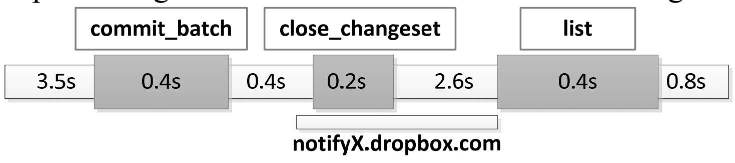
Besides, in this period, it is CPU-cost stage for Dropbox is busy in indexing the data, checking for deduplication and so on.

5.2 The Server Processing

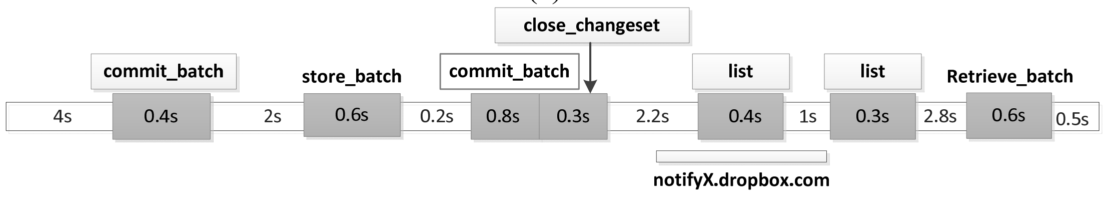
Besides the startup time, the server processing time, which starts from the *close_changeset* connection to the *list* connection, is also an important part in the total processing time. We imply that this may be because the Dropbox server (especially the control server) spends some time to ask the notify server to response corresponding connections. And again it seems that the time cost by syncing small file is a little longer than that of larger file. The reason is unclear as little is known about the Dropbox control server, but probably related to the strategy taken by Dropbox.

5.3 The Time Overhead Before Data Transferring

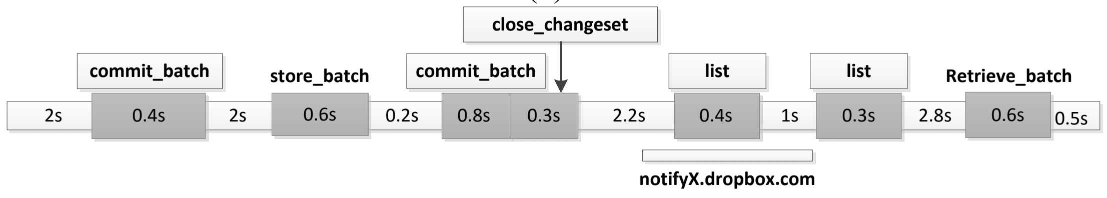
The third part of the processing time is the time before data transferring. As showed in Fig. 2.



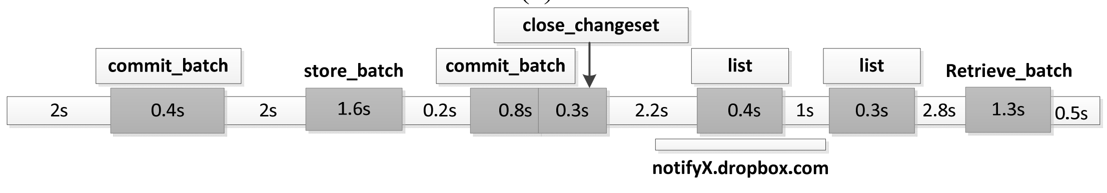
(a) 0KB



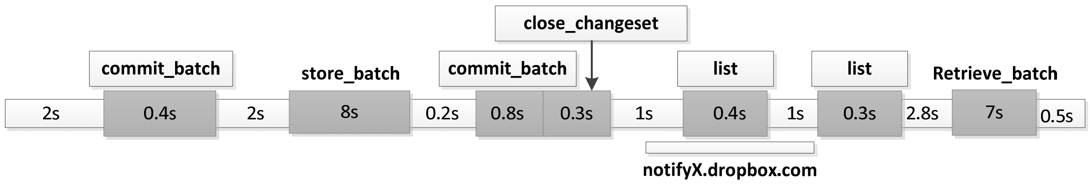
(b) 1KB



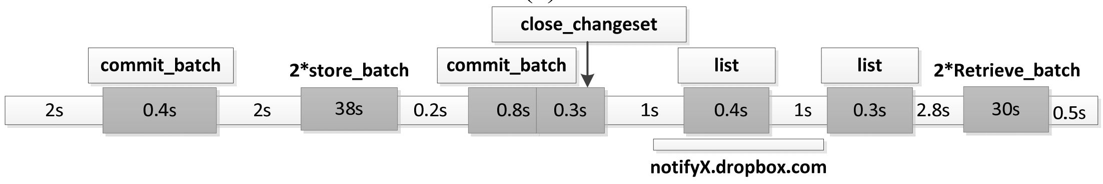
(c) 10KB



(d) 100KB



(e) 1MB



(f) 5MB

Fig. 2 the time overhead inside the procedure with different file size

Both the time before *commit_batch* connection and the time before *retrieve_batch* connection is about 2 seconds or more, no matter what the size of file is. As the chunking or bundling has been done by the startup stage, the probable reason here will be that Dropbox must obtain the locks (like read-write lock) for the file that has been changed, which is obviously time waste.

Above all, the processing time is related to not only from the client by different tasks but also from the server which is confidential. It is obvious that the time overhead of the three part occupies about 80 percent of the total processing time. The system designers should pay more attention to these parts to improve the performance.

6. The Time Overhead of Network

6.1 Data Flow

The data flow is referred to the store_batch and retrieve_batch connections, which are used for uploading and downloading data to/from the storage server.

In theory, the time spent on the uploading is equal to that spent on the downloading. However, In Fig. 2, it is clear that Dropbox spends more time on uploading data than on retrieving data from the cloud, especially for the file with size of 1MB and 5MB. As mentioned above, this is common when users use Dropbox by a proxy server which almost all of them is a virtual machine rent from the cloud server providers with unlimited upload bandwidth and limited download bandwidth. For files smaller than 1MB, this is not obvious because the RTT instead of the data transferring delay is the leading role. As a result, it could be a good choice for syncing small important files between different devices or users, but may be not a good choice for large files with size much larger than the bandwidth of the proxy server. After all, the bandwidth is very expensive and cannot be used efficiently for a user, except that the user has large amount of data storage and retrieval every day.

Although Dropbox provides a LAN Dropbox protocol to sync peer to peer to avoid the delay, devices have to be located in the same LAN, which is not practical.

6.2 Control Flow

Different from the data flow, the control flow contains seraval connections to the control server for the meta data management. As showed in Fig. 2, the time overhead is close from 1KB to 5MB, which is about 2~3 seconds. Again the RTT is the leading role. Compared to the data flow, the time is negligible.

As the control flow and the processing time overhead keeps constant among different file size, the time overhead of the data flow will increase with the file size incresing and will be bottleneck. We hope researchers will pay more attention to this part and try our best to improve the performance.

7. Summary

The time overhead is a key metric for the PCS services. In this paper, we have characterized the time overhead inside the data sync procedure of Dropbox by injecting some code to intercept the

SSL communication and a benchmark experiment measurement. In a real network, data flow will suffer a great bottleneck by the bandwidth provided by the proxy service, which makes it take more time for data upload than download. The characters are applicable to those Dropbox-like applications like Seafile, an open source PCS service. We believe that these findings can be an encouragement for those system designers to develop a more cost-effective application.

References

[1]. Drago I, Mellia M, M Munafo M, et al. Inside dropbox: understanding personal cloud storage services[C]//Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012: 481-494.

[2]. Drago I, Bocchi E, Mellia M, et al. Benchmarking personal cloud storage[C]//Proceedings of the 2013 conference on Internet measurement conference. ACM, 2013: 205-212.

[3]. Goncalves G, Drago I, Couto da Silva A P, et al. Modeling the dropbox client behavior[C]// Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014: 1332-1337.

[4]. Li Z, Wilson C, Jiang Z, et al. Efficient batched synchronization in dropbox-like cloud storage services[M]//Middleware 2013. Springer Berlin Heidelberg, 2013: 307-327.

[5]. Li Z, Jin C, Xu T, et al. Towards network-level efficiency for cloud storage services[C]//Proceedings of the 2014 Conference on Internet Measurement Conference. ACM, 2014: 115-128.

[6]. Li Z, Zhang Z L, Dai Y. Coarse-grained cloud synchronization mechanism design may lead to severe traffic overuse[J]. Tsinghua Science and Technology, 2013, 18(3): 286-297.

[7]. Kholia D, Wegrzyn P. Looking Inside the (Drop) Box[C]//WOOT. 2013.

[8]. Cui Y, Lai Z, Wang X, et al. QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services[C]//Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. ACM, 2015: 592-603.

[9]. DynamoRIO Dynamic Instrumentation Tool Platform. http://dynamorio.org/

[10]. Zhang Y, Dragga C, Arpaci-Dusseau A, et al. *-Box: towards reliability and consistency in dropbox-like file synchronization services[C]//Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems. 2013.

[11]. Zhang Y, Dragga C, Arpaci-Dusseau A C, et al. Viewbox: Integrating local file systems with cloud storage services[C]//Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14). 2014: 119-132.

[12]. Inside the magic pocket. https://blogs.dropbox.com/tech/2016/05/inside-the-magic-pocket/