GrandStore: Towards Large-Scale Free Personal Cloud Storage

Li Zhang,
School of Computer Science and Engineering,
Hunan University of Science and Technology,
Xiangtan, Hunan, 411201, P.R. China
zlhncdsy@163.com

Bing Tang
School of Computer Science and Engineering,
Hunan University of Science and Technology,
Xiangtan, Hunan, 411201, P.R. China
zlhncdsy@163.com

Abstract—Personal cloud storage services are gaining popularity, such as SkyDrive, iCloud, Dropbox, etc. All of them provide a certain amount of free storage space for individual users, while the free space is quite limit, and you should upgrade to a paid account to get extra space. Therefore, a new approach is proposed in this paper, that many free personal cloud storage accounts are integrated in order to realize large-scale free personal cloud storage. A prototype system called GrandStore is designed and implemented, which is based on the principle of OAuth protocol and open API. Specifically, after authorized by the owner of account, GrandStore could manage and control the account, so there is no need for complex login any more. Users only need apply for several free cloud storage accounts, and then account authentication credentials are stored in back-end database of GrandStore, which realizes easily enlarging personal free storage space, and managing all storage space in a unique access entry.

Keywords-Cloud storage; GrandStore; OAuth 2.0; Open platform

I. INTRODUCTION

Data explosion is one of the biggest issues facing IT today. The amount of data that organizations store has grown exponentially in the last 10 years. How to store and manage these large-scale data is really a great problem. One solution to this problem is using cloud storage, an infrastructure that provides on-demand online storage services over Internet. Cloud storage is now the new direction of storage technology, which uses virtualized and scalable storage resource pool to provide storage service for users. It is allowed to use all kinds of method to consume cloud storage service through Internet, such as Web, client program and open interface, following the rule of pay-as-you-go. Cloud storage could deliver online services to individuals or companies, including online file hosting, storage and backup. Recently, personal cloud storage services are very popular and attract our attention, such as Microsoft SkyDrive, Apple iCloud, Google Drive, Dropbox, etc. All of them provide free storage space for individuals, as well as file synchronization service, while the free space is quite limit, and you should pay for extra free space.

Since that there are a variety of free personal cloud storage services, basically users should register accounts to use these services. Usually, one user has several accounts, and these accounts belong to different personal cloud storage providers. In this situation, we are confronted with two great problems. First, how to manage multi-accounts of different cloud storage providers using a unique access entry; second, how to obtain more free storage space, since the free space is quite limit.

To tackle these two challenges, in this paper we propose an integrated storage framework that provides large-scale scalable storage by integrating a plenty of personal cloud storage accounts. Integrating the accounts of different cloud storage providers to deliver a unique access interface makes sense and is quite important. In the proposed storage framework, a plenty of personal free accounts are integrated in order to realize large-scale free personal cloud storage. A prototype system called GrandStore is designed and implemented, which is based on the principle of OAuth protocol and open API. Specifically, after authorized by the owner of account, GrandStore could manage and control the account, so there is no need for complex login any more. Users only need apply for many free cloud storage accounts, and then account authentication credentials are stored in back-end database of GrandStore, which realizes easily enlarging personal free storage space.

Compared with other similar systems, GrandStore is different in three ways. First, GrandStore is a scalable and open system, that is to say, you can add dynamically new accounts to GrandStore without disturbing it. Second, if the developers learn SDK provided by a new personal cloud storage providers, this new product can also be added dynamically to GrandStore. Third, GrandStore depends on database to store user's authentication credential so as to avoid account login, therefore it can store a plenty of accounts to obtain large space. Since GrandStore has such good features, it is a promising system that has great practical value.

The rest of the paper is organized as follows. Section 2 surveys personal cloud storage system and OAuth account authorization protocol. Section 3 introduces the architecture of GrandStore prototype system. Section 4 describes the implementation of GrandStore prototype system and the final section offers concluding remarks.

II. BACKGROUND AND RELATED WORK

In this section, we introduce the background knowledge about free personal cloud storage, as well as the comparison of several free personal cloud storage, and also introduce cloud storage open platform and OAuth protocol.

A. *Free Personal Cloud Storage*

Cloud-based services have been introduced in recent years, offering people and enterprises computing and storage capacity on remote data-centers and abstracting away the complexity of hardware management. As one kind of cloud storage, free personal cloud storage has attracted our attention since these years. As the development and popularity of cloud computing, Hadoop Distributed File System (HDFS) has been the first choice to build reliable cloud storage. The comparison of several popular personal cloud storage providers is shown in Table 1, including Amazon Simple Storage Service (S3), Google Drive, Microsoft SkyDrive, Apple iCloud, Dropbox, etc. It is summarized as follows:

- Most of them provide API interface and programming languages support, such as Java, C++, Python, Ruby, C#.
- Most of them provide APIs Client Library for developers.
- Most of them support OAuth 2.0 or 1.1 protocols.
- Most of them provide file synchronization service.
- Most of them provide limited free space for individuals, and there is also file size limit for upload or store. To remove this restriction, you may upgrade to a paid account which will allow you to upload larger files.

B. *Open Platform*

From the survey on current personal cloud storage providers, we found that they follow the same principle of open platform. Personal cloud storage system open platform allows developers to create their applications to use accounts space, without account login. As it can be seen in Fig. 1, the principle of OAuth[1] account authorization in personal cloud storage open platform is described as the following five steps.

- **Step 1:** The developer creates an *application*, usually through web page to give the name and some other basic information.
- **Step 2:** Open platform returns the *application_key* and *application_secret* to developer.
- **Step 3:** Since the application needs the authentication credential of account, the owner of account is guided to input *username* and *password* to apply for *oauth_token*, which is also called *authorization code* or *access ticket*.
- **Step 4:** Open platform returns the *oauth_token.*
- **Step 5:** The developer collects and stores *oauth_token* for further process.

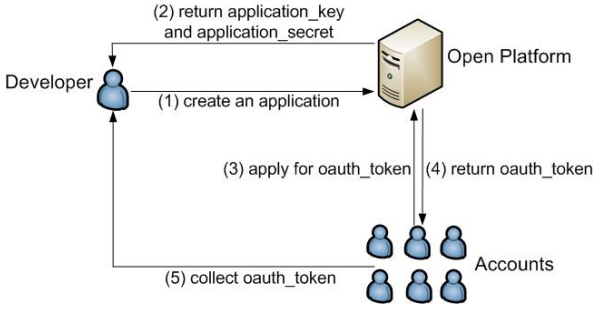

Figure 1. The principle of OAuth-based account authorization cloud storage open platform.

In general, *oauth_token* (or we say *access_ticket*) is composed of two parts, *access_token* and *refresh_token*. Usually, *access_token* has a lifetime, and when it is expired, *refresh_token* is used to generate a new *access_token* and *refresh_token* pair. The expiry period is different, e.g., for some products, it is two weeks; while for some products, it is one month.

C. *Related Work*

Personal cloud storage services are gaining popularity. From the viewpoint of taxonomic, personal cloud storage belongs to the public cloud filed. Many studies have been reported on personal cloud storage or public cloud storage topic in recent years [1][2][3][4][5][6]. For example, Drago et al. [7] studied the characterization of Dropbox, the leading and widely-used personal cloud storage system, and presented a network traffic measurement and analyzed possible performance bottleneck caused by current system architecture and the storage protocol of Dropbox. In [8], the authors presented the architecture for a secure data repository service designed on top of public clouds to support sharing multi-disciplinary scientific datasets.

In [9], the authors examined the efficacy of leveraging web-based email services to build a personal storage cloud, and then presented EMFS, email-based personal cloud storage, which aggregates back-end storage by establishing a RAID-like system on top of virtual email disks formed by email accounts. In particular, by replicating data across accounts from different service providers, highly available storage services can be constructed based on already reliable, cloud-based email storage, while EMFS cannot match the performance of highly optimized distributed file systems with dedicated servers.

The idea of GrandStore is integrating free personal cloud storage accounts. Similarly, DepSky [10] is a system that provides dependable and secure storage in the cloud through the encryption, encoding and replication of the data on diverse clouds that form a cloud-of-clouds. The authors also deployed the system using four commercial clouds to study the performance.

There are also some approaches and systems proposed recently in hybrid storage or integrated storage area. In [11], the authors proposed a scalable, configurable and reliable hybrid storage system, which is composed of stable volunteered personal cloud storage and P2P-based desktop storage system. BitDew [12] is an open source data management middleware for cloud, grid and desktop grid, developed by INRIA, France. It supports using multi-protocols to transfer files. It can be also used as a management tool to distribute/write files to different nodes using FTP, HTTP, and BT protocols. In [13], the authors presented personal storage grid architecture, which provides end-users with web service interface to allow users consume several cloud data space resources, such as online email account space resource and virtual disk space (e.g. FTP service).

[1] http://en.wikipedia.org/wiki/OAuth

Soares et al. [14] presented the FEW Phone File System, a data management system that combines mobile and cloud storage for providing ubiquitous data access. The system takes advantages of the characteristics of mobile phones for storing a replica of a user's personal data to provide high data availability. Defrance et al. [15] presented the view of home networking as a distributed file system, and proposed a solution to organize the home network according to a gateway-centric architecture, where the content access unification for various devices (UPnP/DLNA devices, personal computers, cloud storage systems, etc) is realized at the file system level.

While MetaCDN [16] works in another way, which proposed harnessing storage clouds for high performance content delivery. Several storage clouds are integrated in MetaCDN, providing a unique access interface. The price and bandwidth of storage clouds are considered, which is used for store decision.

III. System Architecture

The objective of GrandStore is to integrate many accounts to obtain large-scale free space, providing a unique access interface. Users firstly register a lot of accounts, integrate them by GrandStore, and then utilize the unique storage space through GrandStore. Users can manage, integrate, and maintain all their own accounts through GrandStore. The system architecture of GrandStore is shown in Fig .2. As you can see, GrandStore is located in the 'middle layer'. The entities in this figure are explained as follows:

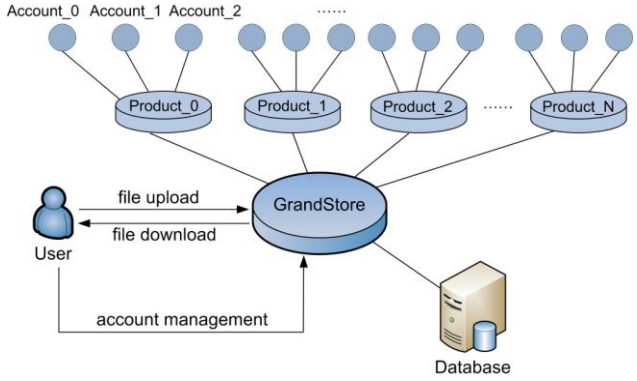

Figure 2. The architecture of GrandStore system.

- Product. It means free personal cloud storage provider, which provides SDK for developers, such as Google Drive, Dropbox and Kuaipan.

- Account. User is allowed to register many accounts for one product. Each account has a limited storage space.

- User. User is responsible for adding dynamically new products and new accounts, and has the right of all kinds of file operations.

- Database. It is used to store account authentication credential (e.g., *access_token*, *refresh_token*) and user's file information (e.g., file name, file size, file path, etc).

GrandStore is based on the principle of OAuth protocol and open API. Specifically, after authorized by the owner of account, GrandStore could manage and control the account, so there is no need for complex login any more. It only needs applying for many free accounts, and then store account authentication credentials to back-end database of GrandStore system, which realizes easily enlarging personal free storage space.

IV. Algorithm and Implementation

GrandStore is released as an open source project, which is developed by Java language, and MySQL 5.0 is selected as the back-end DBMS, and Eclipse is adopted as the development tool. It requires Java SDK provided by personal cloud storage providers. GrandStore now only supports Amazon S3, Google Drive, Dropbox and KingSoft Kuaipan, and it will support more products in the next version.

In this section, we mainly introduce the implementation of core algorithms in GrandStore system. With the aspect of account maintain, we describe account insert algorithm and account authentication credential update algorithm. With the aspect of file operations, we only describe file list, file upload and file download algorithm as three examples. The implementation of other file operations follows the same approach, which is ignored in this paper.

A. Account Insert Algorithm

As we mentioned before, we create one application for each product, and we distinguish them through unique application id. For each product, we can also register many accounts, and each account also has a unique id. The account insert algorithm is shown in Algorithm 1. First, GrandStore starts account authorization guide utility. After authorized by the owner of account through inputting username and password, access token and refresh token are generated, and then stored in database. Using this authentication credentials, there is no need for the owner of account to authorize anymore.

Algorithm 1. Account insert algorithm in GrandStore

Require: Let app_i be the *id* of personal cloud storage product
Require: Let *account<username, password>* be the account to be added
Require: Let acc_j be the *id* of the account to be added
Require: Let *access_token* be the returned access token by open platform OAuth server
Require: Let *refresh_token* be the returned refresh token by open platform OAuth server
Require: Let *creation_time* be the creation time of authentication credential

1: Get app_i that *account<username, password>* belongs to
2: Start account authorization guide utility for product app_i
3: Input *username* and *password* of *account*
4: Login and allow app_i to manage the storage space of *account*
5: Return authentication credential composed of *access_token* and *refresh_token*
6: Get *creation_time* of this authentication credential
7: Write {app_i, acc_j, *access_token*,*refresh_token*, *creation_time*} to database

B. Account Authentication Credential Update Algorithm

Generally, authentication credential has a lifetime, it will became invalid when exceeds expire period. Therefore, we adopt a multi-thread approach to check all accounts, and find those expired access token. Then, the corresponding refresh token is used to generate a new pair of access token and refresh token, supported by OAuth 2.0 protocol. The detailed algorithm is shown in Algorithm 2.

Algorithm 2. Account authorization credential update algorithm in GrandStore

Require: Let *AuthTable*{app_i, acc_j,*access_token*, *refresh_token*, *creation_time*} be the account authentication credential information in database
Require: Let app_i be the *id* of personal cloud storage product
Require: Let acc_j be the *id* of account
Require: Let $lifetime_i$ be the lifetime of authentication credential for product app_i
Require: Let *current_time* be the current time
Require: Let $access\_token_{new}$ be the new access token
Require: Let $refresh\_token_{new}$ be the new refresh token
Require: Let $creation\_time_{new}$ be the creation time of the new authentication credential

1: **for all** record *auth* ∈ *AuthTable* **do**
2: Check $auth.app_i$ and get $lifetime_i$ for this product
3: **if** (*current_time* - *auth.creation_time*) > $lifetime_i$ **then**
4: {authentication credential is expired, use refresh token to get a new one}
5: Check $auth.app_i$ and select corresponding API
6: {create an API session for further API calls}
7: *API.init*($auth.app_i$)
8: *API.create*(*auth.access_token*)
9: {$access\_token_{new}$,$refresh\_token_{new}$}← *API.doRefresh*(*auth.refresh_token*)
10: Get $creation\_time_{new}$ of the new authentication credential
11: {update the authentication credential of account acc_j}
12: Write {app_i, acc_j, $access\_token_{new}$, $refresh\_token_{new}$, $creation\_time_{new}$ } to database
13: **end if**
14: **end for**

C. File List Algorithm

Because GrandStore is designed to integrate many accounts, when the user logs into GrandStore, GrandStore should retrieve and then list all files of each account in a unique file access graphical interface. In order to list all the files from all accounts, it just simply executes API calls to get the files of each account. Algorithm 3 describes the file list algorithm.

Algorithm 3. File list algorithm in GrandStore

Require: Let *AuthTable*{app_i, acc_j, *access_token*,*refresh_token*, *creation_time*} be the account authentication credential information in database

1: {list files of each account}
2: **for all** record *auth* ∈ *AuthTable* **do**
3: Check $auth.app_i$ and select corresponding API
4: {create an API session for further API calls}
5: *API.init*($auth.app_i$)
6: *API.create*(*auth.access_token*)
7: *API.listAllFiles*( )
8: **end for**

D. File Upload Algorithm

When users upload a file to GrandStore, it firstly lookups a proper account to store this file. In this paper, we propose the 'maximal unused space' approach. That is to say, GrandStore lookups the account which has the maximal unused space, and then stores the file to this account. File upload algorithm is demonstrated in Algorithm 4. This approach can achieve storage balance, and avoid the situation that some accounts are too busy than others.

Algorithm 4. File upload algorithm in GrandStore

Require: Let app_i be the *id* of personal cloud storage product
Require: Let acc_j be the *id* of account
Require: Let *AuthTable*{app_i, acc_j, *access_token*, *refresh_token*, *creation_time*} be the account authentication credential information in database
Require: Let *AccountTable*{acc_j, *total_space*, *unused_space*} be the account space consumption information in database
Require: Let *max_space* be a variable to store maximal unused space
Require: Let *opt_account* be the account that has the maximal unused space
Require: Let *F* be the file to be uploaded to the system
Require: Let *fid* be the unique *id* of the file when it is successfully uploaded
Require: Let *FileTable*{*fid*, acc_j} be the file storage mapping information in database

1: {lookup the account that has the maximal unused space}
2: *max_space* ← 0
3: **for all** record *account* ∈ *AccountTable* **do**
4: **if** *account.unused_space* > *max_space* **then**
5: *max_space* ← *account.unused_space*
6: *opt_account* ← *account*
7: **end if**
8: **end for**
9: {lookup the authentication credential of *opt_account*}
10: **for all** record *auth* ∈ *AuthTable* **do**
11: **if** $auth.acc_j$ = = *opt_account* **then**
12: {upload to this account directly}
13: Check $auth.app_i$ and select corresponding API
14: {create an API session for further API calls}
15: *API.init*($auth.app_i$)
16: *API.create*(*auth.access_token*)
17: *fid* ← *API.doUpload*(*F*)
18: {update account space consumption of *opt_account*}
19: *opt_account.unused_space* ← *opt_account.unused_space* - *F.size*( )
20: {insert file storage information}
21: Write {*fid*, $opt\_account.acc_j$} to database
22: **break**
23: **end if**
24: **end for**

E. File Download Algorithm

File download algorithm is relatively easier than file upload algorithm. When users download a file from

GrandStore, it firstly lookups the account that contains this file, and then download from this account directly through API calls. Algorithm 5 indicates file download algorithm.

Algorithm 5. File Download Algorithm in GrandStore

Require: Let *AuthTable*{app_i, acc_j,*access_token*, *refresh_token*, *creation_time*} be the account authentication credential information in database
Require: Let *FileTable*{*fid*, acc_j} be the file storage mapping information in database
Require: Let *F* be the file to be downloaded from the system
Require: Let *store_account* be the account that stores *F*

```
1: {lookup the account that stores F}
2: for all record file ∈ FileTable do
3:   if file.fid = = F.getfid( ) then
4:     store_account ← file.acc_j
5:     break
6:   end if
7: end for
8: {lookup the authentication credential of store_account}
9: for all record auth ∈ AuthTable do
10:  if auth.acc_j = = store_account then
11:    {store in this account, download directly}
12:    Check auth.app_i and select corresponding API
13:    {create an API session for further API calls}
14:    API.init(auth.app_i)
15:    API.create(auth.access_token)
16:    API.doDownload(F)
17:    break
18:  end if
19: end for
```

V. CONCLUSION

Based on the study of current cloud storage system open platforms and OAuth protocol, this paper proposed a method to integrate a plenty of free accounts to get a unify large-scale free personal cloud storage, and also introduced the design and implementation of a prototype system called GrandStore. The core algorithms of GrandStore are described in detail. It is a promising system that has great practical value. First, it proposed a method to get large storage space without upgrading to a paid account. Second, it allows you to manage all your accounts in a unique access interface.

In spite of this, we plan to improve GrandStore in three ways in future work,

- First, we will improve it, and release a new version for Tablet and Android equipment, to manage your own accounts in a mobile terminal.

- Second, we will design optimized algorithms which consider network distance. Generally, free personal cloud storage providers are geographically dispersed, e.g., the server of Google Drive locates in US, while the server of KingSoft Kuaipan locates in China. When users write or read files, selecting the 'closest' provider or account makes sense and very important.

- Third, we will do some I/O performance evaluation for GrandStore, as well as the advantages of network-aware account selection algorithm.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Hunan Province under grant no. 2015JJ3071, as well as the Scientific Research Fund of Hunan Provincial Education Department under grant no. 16C0643 and 12C0121.

REFERENCES

[1] D. Dai, W. Zheng and T. Fan, "Evaluation of personal cloud storage products in China," Industrial Management and Data Systems, Vol 117, Issue 1, 2017, pp. 131-148.

[2] H. Chen, L. Zhang, B. Hu, S. Long and L. Luo, "On Developing and Deploying Large-File Upload Services of Personal Cloud Storage," Proceedings of 2015 IEEE International Conference on Services Computing (SCC 2015), New York City, NY, USA, pp. 371-378.

[3] R. Pitchai, S. Jayashri and J. Raja, "Searchable Encrypted Data File Sharing Method Using Public Cloud Service for Secure Storage in Cloud Computing," Wireless Personal Communications, vol. 90, no. 2, 2016, pp.947-960.

[4] E. Bocchi, I. Drago and M. Mellia, "Personal cloud storage: Usage, performance and impact of terminals," Proceedingd of the 4th IEEE International Conference on Cloud Networking (CloudNet 2015), Niagara Falls, ON, Canada, 2015, pp. 106-111.

[5] K. Ning, Z. Zhou and L. Zhang, "Leverage Personal Cloud Storage Services to Provide Shared Storage for Team Collaboration," Proceedings of IEEE International Conference on Services Computing (SCC 2014), Anchorage, AK, USA, 2014, pp. 613-620.

[6] M. Nebeling, M. Geel, O. Syrotkin, M. C. Norrie, "MUBox: Multi-User Aware Personal Cloud Storage," Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015), Seoul, Republic of Korea, 2015, pp. 1855-1864.

[7] I. Drago, M. Mellia, M. Munafo, A. Sperotto and R. Sadre, A. Pras, "Inside dropbox: understanding personal cloud storage services," Proceedings of the 2012 ACM conference on Internet measurement conference (IMC'12), 2012, pp. 481-494.

[8] A.G. Kumbhare, Y. Simmhan and V. Prasanna, "Designing a secure storage repository for sharing scientific datasets using public clouds," Proceedings of the second international workshop on Data intensive computing in the clouds (DataCloud-SC'11), 2011, pp. 31-40.

[9] J. Srinivasan, W. Wei, X. Ma and T. Yu, "MFS: Email-based Personal Cloud Storage," Proceedings of the 6th IEEE International Conference on Networking, Architecture and Storage (NAS'2011), 2011, pp. 248-257.

[10] A. Bessani, M. Correia, B. Quaresma, F. Andr é and P. Sousa, "DepSky: dependable and secure storage in a cloud-of-clouds," Proceedings of the Sixth European conference on Computer systems (EuroSys 2011), 2011, pp. 31-46.

[11] B. Tang and G. Fedak, "Analysis of data reliability tradeoffs in hybrid distributed storage systems," Proceedings of the 17th IEEE International Workshop on Dependable Parallel, Distributed and Network-Centric Systems (DPDNS 2012), 2012, pp. 1540-1549.

[12] G. Fedak, H. He and F. Cappello, "BitDew: A data management and distribution service with multi-protocol file transfer and metadata abstraction," Journal of Network and Computer Applications, vol. 32, no. 5, 2009, 961-975.

[13] M.-G. Lim, S. Wu, T. Simon, M. Rashid and N. Helian. "Personal Storage Grid Architecture: Consuming Cloud Data Space Resources," International Journal of Grid and High Performance Computing, vol. 2, no. 3, 2010, 17-30.

[14] J. Soares and N. Pregui ça, "Combining Mobile and Cloud Storage for Providing Ubiquitous Data Access," Proceedings of the 17th International Conference on Parallel Processing (Euro-Par 2011), Lecture Notes in Computer Science (LNCS), Volume 6852/2011, Springer-Verlag, 2011, pp. 516-527.

[15] S. Defrance, R. Gendrot, J. Le Roux, G. Straub and T. Tapie, "Home Networking as a Distributed File System View," Proceedings of the 2nd ACM SIGCOMM workshop on Home networks (HomeNets'11), 2011, pp. 67-72.

[16] J. Broberg, R. Buyya and Z. Tari, "MetaCDN: Harnessing 'storage clouds' for high performance content delivery," Journal of Network and Computer Applications, vol. 32, no. 5, 2009, pp. 1012-1022.