A Design of Cloud Storage Gateway Based on Data Dispersal[*]

Hongyang Hu, Di Zhang and Yuezhi Zhou
Department of Computer Science and Technology Tsinghua University, Beijing 100084, China
guyuehongyang@163.com zd0929@gmail.com zhouyz@mail.tsinghua.edu.cn

Abstract - A cloud storage gateway is a network appliance or server which resides at the customer premises. It can integrate and translate different cloud storage APIs and increase data security. A variety of designs of cloud storage gateway have been proposed. However, existing designs cannot help users to access data stored on the cloud when cloud storage cannot be reached due to maintenance, interruptions or other failures. In this paper, we present a design of cloud storage gateway based on data dispersal. In the cloud storage gateway of our design, data is divided into redundant data blocks and parity blocks, and these blocks are stored into different cloud storage respectively. When a portion of cloud storage cannot be accessed, the new designed cloud storage gateway still able to access the required data by restoring data from other available cloud storage. Further, if an acceptable number of data blocks are lost or erroneous, the cloud storage gateway of our design can also restore the original data. And the gateway utilizes an improved LRU to improve the efficiency of cache. Experimental results show that the cloud storage gateway of our design can restore the original data when some cloud storage fail and improves the access efficiency by 20%.

Key Words - cloud storage gateway; data dispersal; data division; data restoration

1. Introduction

Since the introduction of cloud computing technology in 2006, it has developed rapidly over the years. With the continuing upgrade of calculation model, storage technology which is an inseparable with calculation is developed from stand-alone storage, network storage, and distributed storage to cloud storage [1]. Cloud storage makes a large number of different types of storage devices to work together through virtualization software and provides data storage service to users [2]. Compared with traditional storage, cloud storage is device-independent and cost-effective. Therefore, it is increasingly widely used in various fields.

However, most public cloud storages rely on Internet protocols, using a REST API [3] over HTTP, instead of using a conventional SAN (Storage Area Network) or NAS (Network Attached Storage) protocol [4]. This results in incompatibilities between cloud storage and legacy applications, which make troubles to the users. To solve this problem, Cirtas, TwinStrata, Amazon and some other enterprise launches cloud storage gateway, which help users to access to cloud storage just like accessing their local disk.

Cloud storage gateway which resides at the customer premises, serves as intermediary between local applications and multiple cloud storage providers [5]. Cloud storage gateway provides basic protocol conversion which makes incompatible technologies communicate with each other simply and makes cloud storage works as a NAS filter, a block storage array, a backup target or even an extension of applications. This provides a seamless integration with existing applications. Additionally, cloud storage gateway includes features such as data compression, data encryption, de-duplication, snapshots and version control.

Cloud storage gateway can be classified into two categories. One is cloud storage gateway provided by cloud storage service provider, such as Amazon AWS storage gateway [6]. The other is cloud storage gateway launched by a third party, which integrates cloud storage APIs of multiple cloud storage service providers. With this category of cloud storage gateway, users can choose multiple different cloud storage providers according to their needs. Typical representatives of third party storage gateway are cloud storage gateway launched by Cirtas, Nasuni, StorSimple, TwinStrata, and so on.

When a cloud storage service cannot be reached due to maintenance, interruptions or other failures, existing cloud storage gateway cannot help users to access to their data stored on the cloud storage. Worse, when data stored in the cloud storage are lost or damaged, the existing two categories of cloud storage gateway cannot help users to recover data.

In addition, cloud storage gateway provides a certain size (1T to 8T) of cache, in which users can store frequently accessed data. However, existing cloud storage gateway requires users to manually set what data are cached. Users have to manually clear the cache to free up space to store new data. Further, cloud storage gateway often provides service for multiple users. Therefore maintaining cache becomes a tedious work for users.

To overcome the deficiencies of existing cloud storage gateway, we propose a new design of cloud storage gateway based on data dispersal. Data are divided into redundant data blocks and parity blocks through the use of segmentation and recovery technologies, and these data blocks are stored into different cloud storage respectively. Compared with existing designs of cloud storage gateway, the main contribution of our design can be summarized as follows:

(1) The availability of cloud storage service is enhanced by cloud storage gateway of our design. When a portion of cloud storage services cannot be accessed, the new designed cloud

[*] This work is supported by National 863 Project(No. 2011AA01A203) and National Science & Technology Pillar Program(No.2012BAH13F04)

storage gateway still able to access the required data by restoring data from other available cloud storage services.

(2) The data reliability stored in cloud storage service is also enhanced. If an acceptable number of data blocks are lost or erroneous, the cloud storage gateway of our design can also restore the original data.

(3) We use the improved LRU (Least Recently Used) algorithm [7] to manage the cache space of cloud storage gateway of our design. The improved LRU algorithm can realize automatic management of cloud storage gateway cache space, which improves the efficiency of data access and enhances the user experience.

2. Architecture

The cloud storage gateway presented in this paper works as a bridge between users and cloud storage service provider. It allows users to access cloud storage as easy as to their local disks. As the Figure 1 shows, this cloud storage gateway includes data compression and decompression module, data segmentation and recovery module, data encryption and decryption module, cache module, API integration module.

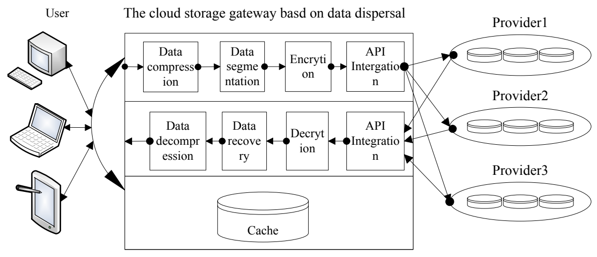

Figure 1 the cloud storage gateway based on data dispersal

2.1. Data Compression and Decompression Module

When users would like to store their data, this module will compress the users' data which may contain a large number of files to save storage space so that users can reduce the cost of cloud storage. This module can also enhance the efficiency of data segmentation and recovery module. This module will help users decompress their data when they want to access to their data through cloud storage gateway presented in this paper.

This module is implemented with the LZ4 [8] which is an open source compression algorithm. LZ4 is a very fast lossless compression algorithm that can reach the compression rate of 300MB/s and the decompression rate of 1GB/s with single-core CPU. LZ4 also supports multi-core CPU.

2.2. Data Segmentation and Recovery Module

This module is a core part of the cloud storage gateway proposed in the paper. When users would like to store their data, this module will divide the data into several blocks and redundantly encode some new blocks. Then the gateway put these blocks into different cloud storage service provider. Each provider only store one block. This module is able to recover original data for users even if some of these providers are not available.

When users want to store their data, this module first divides the compressed user data into *n* equal-length data blocks, symbol for $d_1, d_2, d_3 \ldots d_n$. Then this module generates *m* parity blocks by the *n* equal-length data blocks, symbol for $c_1, c_2, c_3 \ldots c_n$. The size of parity blocks is equal to that of data blocks. This module first gets the size of data, symbol for *Size*, and divides the data into *n* equal-length data blocks. The size of each block is $\lceil Size/n \rceil$. The working principle shows as the follows.

When users would like to store their data, this module works as follow steps:

1) This module constructs a matrix *A* as follow:

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & n^{m-1} \end{bmatrix}$$

2) The module divides the user data into *n* equal-length data blocks and means the data blocks $d_1, d_2, d_3 \ldots d_n$ in a matrix *D* as follow:

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}.$$

3) The module generates *m* parity blocks as the follow method:

$$\underbrace{\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}}_{E} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & n^{m-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}}_{D}$$

Matrix E is a set of n data blocks and *m* parity blocks. The user data have already been successfully divided and encoded into *n* data blocks and *m* parity blocks now. All the blocks are equal-length with each other.

4) This module gives every data block and parity block a unique name and put them into *m*+*n* cloud storage service providers. Each provider only has one data block or parity block.

Assuming the case that as many as *m* data blocks and parity blocks cannot be accessed when users want to read their data, this module has to recover original user data by the rest *n* data blocks and parity blocks. If the n blocks are all data blocks, it is easy to recover the data directly. If not, this module works as the following steps:

1) This module constructs an $n*1$ matrix E' with the n data blocks and parity blocks.
2) This module chooses n rows of matrix A which the n data blocks and parity blocks correspond to and constructs an $n*n$ non-singular matrix A'.
3) A' is a non-singular matrix so that we can calculate its inverse matrix A'^{-1}. A'^{-1} is an n*n matrix.
4) The original data $D=\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}=A'^{-1}*E'$.

The principle described above will be shown as the follow example in Figure 2. The original user data are divided into 4 data blocks and encoded into 2 parity blocks. All the 6 blocks are equal-length. They are stored into 6 cloud storage service providers. Each provider only has one of them. Assuming that there are 2 providers cannot be accesse, the gateway presented in this paper will recover the original user data by the rest 4 blocks with the help of data segmentation and recovery module.

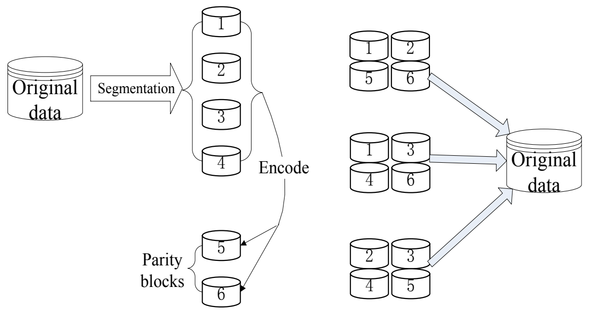

Figure 2 Data segmentation and recovery schematic

2.3. Data Encryption and Decryption Module

To improve the security of user data, the paper designs a data encryption and decryption module for the gateway. With this module, no one can get the original content of the data from the encrypted data. This module encrypts user data when users want to store their data and decrypts them when users would like to access to their data.

This module utilizes ElGamal encryption algorithm [9] which is an open-source algorithm to accomplish the encryption work. ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key exchange [10].

2.4. Cache Module

It is tedious work for users to always clean up the gateway cache. This paper improved the existing LRU (Least Recently Used) cache algorithm to cater to the need of the users. The gateway presented in this paper utilized the improve LRU cache algorithm to automatically the content of its cache module. The users will have a good experience when they store or access to their data by this gateway.

The improved LRU cache algorithm is implemented as follows.

1) The module sets up some user-spaces whose size must be more than a specific set or a minimum set. The size of user-spaces can be changed dynamically. Every user has one. Thus, users can always get their frequently accessed data quickly even if they left off in the past period of time. User-space will be deleted if its user leaves. It will be created when new user joins. This design follows the fairness principle for users.
2) When a user would like to access to their data, the gateway proposed in this paper first searches the data in the cache. If the user data are in the cache, the gateway sends it to users directly. Then the user data will be assigned with the highest priority in the user-space. And the user-space that stores the user data is assigned the highest priority in the cache.
3) When the data which a user would like to access are not in the gateway cache, the gateway puts the data into the cache after data segmentation and recovery module recovers the data for user. If the cache has enough free space, the gateway will put the data into the space of the user and assign the data the highest priority in the user space and assign the user space the highest priority in the cache. If the free space of the cache is insufficient, the gateway will eliminate the lowest priority data in the lowest priority user space at first. It must be guaranteed that the size of all the data in the lowest priority user space is not less than the minimum set. Otherwise, the gateway will assign this user space the highest priority.
4) The gateway repeats the step 3) until the data are put into the cache successfully.
5) In general, users can afford the time cost on storage and read for large data, while they will not tolerate that for small data. And the cache space of the gateway is limited. So the cache prefers the data which are smaller than 500MB. If users would like to put the data more than 500MB into the cache, they have to do it manually.

2.5. API Integration Module

This module simplifies the use of cloud storage. It makes cloud storage work as a local disk. Users do not need to program to use the cloud storage. In order to put the data blocks and parity blocks into different cloud storage service providers and get these blocks from these provider, this module integrates the storage API and read API of several cloud storage service provider, including Baidu cloud storage, Alibaba cloud storage, SNDA cloud storage and Google cloud storage. For example, the storage API of Alibaba cloud storage is *put_object_from_file()*, and the read API is *get_object_to_file()*.

2.6. Backup mechanism

In the present gateway system, the reliability of the gateway itself must be considered. As shown in Figure 3, the gateway system contains a main gateway and a standby gateway. They have the same deployment. The standby gateway always backups the content of the main gateway after every operation on the main gateway and it is kept the same status with the main one. The standby one will replace the main one when the main one fails. The backup mechanism improved the reliability of the gateway system.

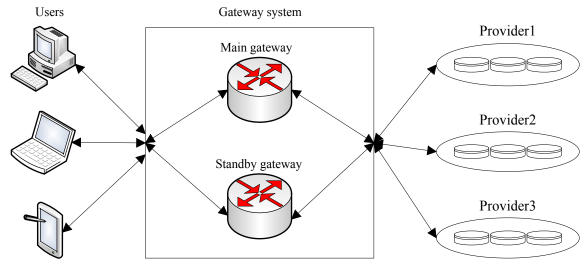

Figure 3 Gateway system schematic

3. Evaluation

The gateway prototype is implemented on two same machines that are located in a LAN. One is the main gateway, the other one is the standby gateway. The configuration of the gateway is showed by the following table 1.

Table 1 The configuration of the gateway

CPU	Intel Xeon E3110 Dual-core 3.0GHz
Memory	DDR2 4.0GB
Hard Disk	SATA 250GB
Network Adapter	Broadcom NetXtreme II 5722 Gigabit Ethernet
Operating System	Linux version 3.6.7

The users are eleven members in my lab. The user data include video, audio, documents, pictures, tables and other types of files.

In the experiment, the gateway divided the data a user would like to store into three data blocks and encoded into 1 parity block and respectively put them into Baidu cloud storage, Alibaba cloud storage, SNDA cloud storage and Google cloud storage. Each provider only had one data block or parity block. We make hundreds of experiments about the data recovery when one of the four providers cannot be accessed. Experiment showed that, the gateway could recover user data successfully when anyone of four providers cannot be accessed.

The gateway provides 5GB minimum user space and 200GB maximum user space which is limited by the hard disk storage size. Because of the implementation of improved LRU cache algorithm, the gateway automatically maintained the content of the cache successfully. Only a small number of large data were decided whether they would be put into cache by users themselves. It improved the user experience. In the experiment, these eleven users read types of data about 2000 times. Experiment showed that this gateway increased the reading efficiency by 20% compared with TwinStrata cloud storage gateway and Nasuni cloud storage gateway.

4. Conclusion

A cloud storage gateway is a network appliance or server which resides at the customer premises and connects local applications and multiple cloud storage services. Existing cloud storage gateway cannot help users access data stored on the cloud when cloud storage cannot be reached due to maintenance, interruptions or other failures. In this paper, we propose a new design of cloud storage gateway based on data dispersal. The availability of cloud storage service and the data reliability stored in cloud storage service are enhanced through the use of segmentation and recovery technologies. In addition, the improved LRU algorithm is used to automatically manage the cloud storage gateway cache space, which improves the efficiency of data access and enhances the user experience.

5. Future Work

The number of integrated cloud storage APIs in the prototype of the new designed cloud storage gateway is relatively small. We will develop and integrate more cloud storage APIs in the future. Further, because different cloud storage service providers have different characteristics and charges, we will design an optimal algorithm for users to automatically select the appropriate cloud storage services.

Acknowledgment

This work is partially supported by the National 863 Project (No. 2011AA01A203) and the National Science & Technology Pillar Program (No.2012BAH13F04).

References

[1] Lingdi Ping, Xiaoping Ge, Ya Wang, Jiangqing Fu. Cloud Storage as the Infrastructure of Cloud Computing.Intelligent Computing and Cognitive Informatics (ICICCI),2010. pp. 380-383.
[2] GigaOm, Show Me the Gateway — Taking Storage to the Cloud, By Gary Orenstein Jun. 22, 2010.
[3] John Bresnahan, Kate Keahey, David LaBissoniere, Tim Freeman. Cumulus: an open source storage cloud for science. ScienceCloud'11 Proceedings of the 2nd international workshop on Scientific cloud computing, 2011. Pages 25-32.
[4] M Padovano. System and method for accessing a storage area network as network attached storage. US Patent 6606690, 2003
[5] Anand Prahlad, Marcus S.Muller, Rajiv Kottomtharayil, Srinivas Kavuri, Parag Gokhale, Manoj Vijayan. Cloud gateways system for managing data storage to cloud storage sites. US Patent 0333116. Dec.30,2010.
[6] AWS Storage Gateway. http://aws.amazon.com/cn/storagegateway/.
[7] Clarence W. DeKarske, St. Paul Park, Minn. Paired least recently used block replacement system. US Patent 4168541. Sep.18, 1979
[8] Pan, T. & Uhlenbeck, O.C. A small metalloribozyme with a two-step mechanism. Nature 1992. 358(6387):560-563.
[9] ElGamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Advances in Cryptology-Crypto'84, LNCS 196, pp.10-18, Springer-Verlag, 1984.
[10] Diffie W, Hellman M. New directions in cryptography. IEEE Transactions on Information Theory 1976. 22(6):644-654.