T-CloudDisk: A Tunable Cloud Storage Service for Flexible Batched Synchronization

Zhenhua Li [1], He Xiao [1], Linsong Cheng [1], Zhen Lu [1], Jian Li [1],
Christo Wilson [2], Yao Liu [3], Yunhao Liu [1], Yafei Dai [4]
[1] Tsinghua University [2] Northeastern University [3] Binghamton University [4] Peking University
lizhenhua1983@tsinghua.edu.cn, cbw@ccs.neu.edu, yaoliu@cs.binghamton.edu

ABSTRACT

Cloud storage services such as Dropbox have quickly gained enormous popularity in recent years. They offer users with convenient and reliable approaches to store and share data from anywhere, any device at anytime. However, they are still suffering from the *"traffic overuse problem"* in the presence of *frequent, short* data updates [3]. To address this problem, we are implementing a *tunable* cloud storage service (named "T-CloudDisk") for *flexible* batched synchronization. This paper introduces the characteristics, technical approach, and preliminary timeline of T-CloudDisk.

Categories and Subject Descriptors

H.2.4 [**Information systems**]: Information storage systems—*Storage architectures, Cloud based storage*

General Terms

Design, Performance

Keywords

Cloud storage, Traffic overuse, Batched synchronization

1. INTRODUCTION

Cloud storage services such as Dropbox, SkyDrive, and Google Drive have quickly gained enormous popularity in recent years. They offer users with convenient and reliable approaches to store and share data from anywhere, any device at anytime. However, we observe that cloud storage services are still suffering from the *"traffic overuse problem"* in the presence of *frequent, short* data updates, and thus propose a middleware-based batched synchronization solution (named "UDS", *i.e.* Update-batched Delayed Sync) to address this problem [3].

Although UDS significantly reduces the traffic overuse, acting as a middleware between the user's local filesystem and the concerned cloud storage application (like Dropbox), it requires considerable additional storage space in the user's local disk, and currently it uses fixed *buffer size* and *timer threshold* for batched synchronization.

Middleware 2013 Posters and Demos Track, December 9–13, Beijing, China

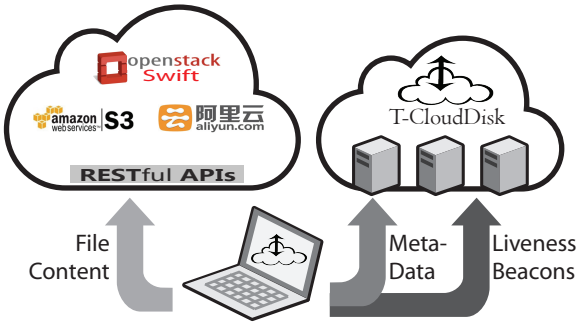

Figure 1: High-level system architecture of the T-CloudDisk storage service.

In order to overcome these shortcomings, we are implementing an *independent*, *tunable* cloud storage service (named "T-CloudDisk") for *flexible* batched synchronization. The abovementioned three *characteristics* are explained as follows:

- *Independent*: T-CloudDisk is a self-contained cloud storage service, rather than a middleware that relies on other services' applications. As depicted in Figure 1, T-CloudDisk has its own user application and private cloud that maintains the critical meta-data and user liveness information. With such an independent service, we are able to conduct more in-depth, white-box research on cloud storage.
- *Tunable*: To avoid the expensive infrastructure cost and enhance the service scalability, T-CloudDisk outsources all the file contents to a (few) public cloud(s) that support RESTful APIs (in particular the *bucket-object* data model), such as Amazon S3, Aliyun.com OSS, and so forth. Besides, we are also deploying our own "public cloud" based on the open-source OpenStack Swift framework.
- *Flexible*: The T-CloudDisk application allows the user to customize the buffer size and timer threshold according to her/his specific requirements, as shown in Figure 2. This increases the flexibility of batched synchronization because it enables the user's own tradeoff between traffic usage and user experience.

In the remainder of this paper, we first review the related work in Section 2, and then briefly describe our technical approach as well as a preliminary timeline in implementing the T-CloudDisk service.

2. RELATED WORK

Among today's dozens of commercial cloud storage services, Dropbox is the earliest and most representative. Drago *et al.* s-

tudy the system architecture and performance of Dropbox through both large-scale ISP-level measurements [2] and small-scale benchmarking experiments [1]. Their results reveal that frequent, short data updates may well impair the data sync throughput and traffic efficiency, and suggest using a bundling scheme for batched synchronization.

Li *et al.* are the first to (explicitly) identify the "traffic overuse problem" that causes Dropbox-like cloud storage applications to upload unnecessarily large amounts of data sync traffic to the cloud [3, 4]. To address this problem, they propose an efficient batched synchronization middleware solution (*i.e.* UDS) in [3], and an adaptive timer-triggered delta sync solution (*i.e.* aTDS) in [4], respectively. The T-CloudDisk service presented in this paper can be taken as an extension work of UDS and aTDS.

3. APPROACH AND TIMELINE

3.1 Technical Approach

Implementing a Dropbox-like cloud storage service (such as T-CloudDisk) typically relies on two fundamental utilities: 1) a *file change detector* that monitors changes to the user's local sync folder and reports them to the user's application; 2) a *delta sync utility* that computes the binary diff of modified files and only sends the altered bits to the cloud (for saving sync traffic). As for Linux, the former can be the kernel inotify API and the latter can be the standard rsync tool. For example, the UDS middleware [3] is developed by using inotify and rsync. As for Windows, inotify can be replaced by the FindFirstChangeNotification and ReadDirectoryChangesW APIs, and rsync can be replaced by the cwRsync tool. In a word, the technical approach is independent of operating systems.

Flexible batched synchronization of T-CloudDisk is mainly enabled by two critical variables: 1) the buffer size c, *i.e.* the "Batched Sync Buffer" in Figure 2, and 2) the timer threshold t, *i.e.* the "Batched Sync Timer" in Figure 2. Specifically, the cloud storage application maintains a byte counter that ensures frequent, short updates are batched together into chunks of at least some minimum size (*i.e.* c bytes) before they get pushed to the *cloud* (including both the public cloud and the private cloud). Meanwhile, the application also maintains a timer. Whenever a file is created/modified, the timer gets reset to zero. Once the timer reaches a threshold value (*i.e.* t seconds), then all new/modified files in the sync folder are pushed to the cloud, regardless of how much the byte counter accounts to. In essence, the two variables represent the *tradeoff* between sync traffic usage and timeliness of updates to the cloud (or says user experience).

When configuring the two variables for the UDS middleware, we chose $c = 250$ KB and $t = 5$ seconds based on our Dropbox benchmarking experiments [3], because using a larger c or t only brings about trivial sync traffic decrease while obviously affects the user experience. Nevertheless, the above fixed settings of c and t are generally restricted to Dropbox [1] and our specific benchmarking environments. As a result, we allow users to customize their own buffer size and timer threshold in T-CloudDisk, so as to adapt to their diverse working requirements and environments.

3.2 Preliminary Timeline

At the moment (Oct. 10, 2013), we have implemented the following functions for T-CloudDisk on top of the Aliyun.com OSS (Online Storage Service):

[1]Even Dropbox may adjust its system architecture and data sync mechanism in the future.

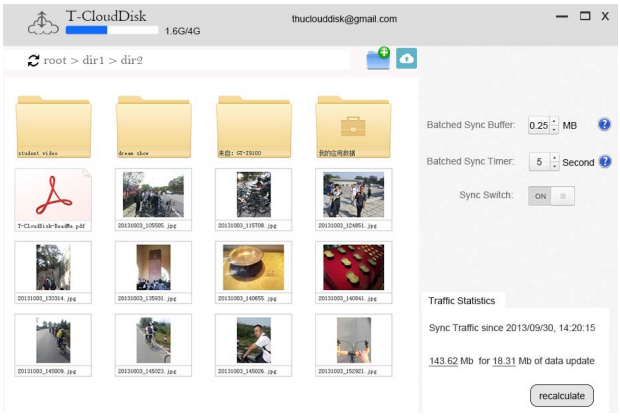

Figure 2: UI snapshot of the T-CloudDisk PC client.

- User account registration and log on.
- Basic file (folder) upload, download, modify, and delete operations.
- Batched synchronization with regard to a single file.
- The control panel for configuring the buffer size c and timer threshold t.
- The statistics panel for displaying how much traffic is consumed for synchronizing how much data updates.

Below are what we expect to finish before the Middleware'13 live demo date (Dec. 9, 2013):

- Batched synchronization across multiple files.
- Novel file and data updates compression before synchronization.
- T-CloudDisk can work on top of other public clouds than Aliyun.com OSS, such as Amazon S3 (Simple Storage Service) and OpenStack Swift.

Finally, our long-term plan includes (but is not limited to):

- Homepage hosting, flexible file sharing, and collaborative file editing functions.
- Full-file and block-level file deduplication.
- File version management, in particular file version rollback for data recovery.
- Developing mobile (*e.g.* Android and iOS) applications for T-CloudDisk.

4. REFERENCES

[1] DRAGO, I., BOCCHI, E., MELLIA, M., SLATMAN, H., AND PRAS, A. Benchmarking Personal Cloud Storage. In *Proc. of IMC* (2013).

[2] DRAGO, I., MELLIA, M., MUNAFÒ, M., SPEROTTO, A., SADRE, R., AND PRAS, A. Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proc. of IMC* (2012).

[3] LI, Z., WILSON, C., , JIANG, Z., LIU, Y., ZHAO, B., JIN, C., ZHANG, Z.-L., AND DAI, Y. Efficient Batched Synchronization in Dropbox-like Cloud Storage Services. In *Proc. of Middleware* (2013).

[4] LI, Z., ZHANG, Z.-L., AND DAI, Y. Coarse-grained Cloud Synchronization Mechanism Design May Lead to Severe Traffic Overuse. *Journal of Tsinghua Science and Technology 18*, 3 (2013).