REVIEW OF MOBILE CLOUD STORAGE SERVICES USING QUICK SYNC

[1]Sangeetha N A, [2]Vivek Sharma S

Department of CSE, AIET, Mijar
Karnataka, India

***Abstract*: The inefficiency problem of modern mobile cloud storage services is identified here. Proper use of bandwidth is not done by existing synchronization and as well the large amount of unnecessary sync traffic is generated even after the implementation of incremental sync. To improve the sync efficiency for mobile cloud storage services, and build the system on two commercial sync services QuickSync (Quick Synchronization) is introduced, QuickSync is a system with three novel techniques. By experimental results QuickSync is able to reduce up to 73.1% sync time in particular experimental settings.**

1. INTRODUCTION

The popular services are Dropbox and Seafile by which many companies have been attracted to enter this market and have offered their own mobile cloud storage services. Network communications is used to update local file changes in cloud. One of the most important performance metrics in cloud storage services is sync efficiency which is used to update local file changes, which is expected to be quickly synchronized to the cloud and then to other devices with low traffic overhead.

Mobile devices are increasing with new demands of ubiquitous storage to synchronize users personal data from anywhere at any time and with any connectivity. Some cloud storage providers have extended and deployed their services in mobile environments, with optionally implemented functions to improve the transmission performance such as chunking and deduplication.

closed source with data encrypted is commercial storage services, whose designs and operational processes remain unclear to the public. So it's hard to understand. Second, some existing services try to improve the sync performance by incorporating several capabilities, it is still unknown whether these capabilities are useful or enough for good storage performance in mobile/wireless environments. Finally, both storage and network fields are used to involve techniques in cloud storage system, it requires the storage techniques to be adaptive and work efficiently in the mobile environment where the mobility and varying channel conditions make the communications subject to high delay or interruption.

The set of techniques to increase the sync efficiency in modern cloud storage is identified, analyzed and proposed to address above challenges consists of three major components: 1) identifying the performance of popular commercial cloud storage services in mobile/wireless environment, 2) analyzing the problems in detail, and 3) proposing a new mobile cloud storage system to enable efficient sync operations in mobile cloud storage services.

The sync performance of the most popular commercial cloud storage services in mobile/wireless networks is measured. The results show that the sync protocol used by these services is inefficient. The sync protocol cannot utilize full available bandwidth in high RTT environment or when synchronizing multiple small files. For example, a minor document editing process in dropbox may result in sync traffic 10 times that of the modification.

Decryption is done to identify the root cause of inefficiency and the trace of data in sync protocol is analyzed. The factors for inefficiency are the inherent limitations of the sync protocol and the distribution storage architecture is the practically implementation of the delta encoding algorithm. The failure in the incremental sync may lead to a large traffic overhead. QuickSync, a system with three novel techniques to improve the sync efficiency for mobile cloud storage services is proposed based on some observations and analyzations to reduce sync time. Network-aware chunker is used to select the proper chunking strategy based on real-time network conditions. Redundancy eliminator to correctly perform delta encoding between two similar chunks located in the original and modified files at anytime during sync process. to improve the network utilization of sync protocol and reduce the overhead when resuming the sync from an interruption. All these are used to reduce the sync time.

QuickSync system is built on dropbox which is the most popular cloud storage services, and seafile which is a popular open source personal cloud storage system. These techniques are used to improve the sync latency for cloud storage services. Results show that there is a improvement in the sync efficiency, reducing up to 73.1% sync time using QuickSync.

2. Synchronization Inefficiency

The series of experiments to investigate the sync inefficiency issues existing in four most popular commercial cloud storage service systems in mobile/wireless environments.

2.1 Architecture and Capabilities

Data sync is the key operation of the cloud storage services, which automatically maps the changes in local file systems to the cloud via a series of network communications. Typical architecture of cloud storage services and the key capabilities are often implemented for speeding up data transmissions.

Architecture: A typical architecture of cloud storage services consists of three major components: the client, the control server and the data storage server.

Typically, user has a designated local folder (called sync folder) where every file operation is informed and synchronized to the cloud by client. The client splits file contents into chunks and indexes them to generate the metadata (including the hashes, modifies time etc). Metadata and contents of user files are separated and stored in the control and data storage servers. The metadata are exchanged with the control server through the metadata information flow, while the contents are transferred via the data storage flow during the sync process. The control server and the data storage server may be deployed in different locations in a practical implementation. Notification flow pushes notifications to the client once changes from other devices are updated to the cloud.

Key capabilities: It uses the several capabilities to optimize the storage usage and speed up data transmissions.

2.2 Low DER Not Equal to Efficiency

DER (Deduplication Efficiency ratio) is defined as the ratio of the deduplicated file size to the original file size which is used to evaluate the effectiveness of deduplication in reducing the original transmission data size. Removing of more redundancy is called as lower DER in which the total sync time can be reduced. Lower DER will not always make sync efficient.

Wireshark is used to measure the packet level trace of the two services in a controlled Wi-Fi environment to study the relationship between the sync time and DER. According to the typical RTT values in mobile/wireless network we use tc to tune the RTT for each service. Most services did not implement the deduplication on the android platform so we only perform measurements on windows platform.

2.3 Failure of Incremental Sync

Some services such as Dropbox leverage is used to achieve incremental sync instead of full-file sync to reduce the network traffic for synchronizing changes. The client software may synchronize much more data than expected and the incremental sync is not always available. Traffic utilization overhead (TUO) is incurred.

Firstly, metadata and contents are completely updated to the cloud with synchronized files to perform all operations. In typical real world usage patterns we perform three types of basic operation: flip bits, insert and delete several continuous bytes at the head, end or random position of the test file.

In second experiment, when the sync data are in the middle of transmissions to the cloud we investigate the sync traffic of performing the modification on the files. After creating a file a flip operation is performed immediately. MS-word is an application which creates fresh temp files and periodically modifies them at runtime. The traffic under different network conditions are seen by involving additional RTT with the help of tc. The result of sea file is same that of the dropbox. is introduced to characterize the network usage, which is defined as a ratio of the practical measured throughput to the theoretical TCP bandwidth. To evaluate how well the cloud storage service can utilize the available network bandwidth to reduce the sync time we apply BUE, whose value is between 0 and 1.

3 Root Cause of Sync Inefficiency

3.1 Pinning Down the Sync Protocol

The spilt file is indexed locally in the sync preparation stage, and chunks are identified in the block list which is sent to the control server in the commit_batch. Through hash-based checking we can identify the chunks existing in the cloud and only new chunks will be uploaded. The client communicates with the storage server directly in the data-synchronization. In each iteration several chunks will be sent and the client synchronizes data iteratively. The client updates the metadata through the list message to inform the server that a batch of chunks have been successfully synchronized at the end of one round of iteration and the server sends an OK message in response. The client communicates with the control server again to ensure that all chunks are updated by the commit_batch, and updates the metadata in the sync-finish stage.

4 System Design

QuickSync, a novel system which concurrently exploits a set of techniques over current mobile cloud storage services to improve the sync efficiency.

4.1 System Overview

It consists of 3 key components: the Network aware chunker, to identify the redundant data before the sync process to improve the sync efficiency. Where the deduplication techniques focus on saving the storage spaces, improving the efficiency for large scale remote backup, or only optimizing the downlink object delivery Cloud storage involve huge overhead and require an important property named "stream-informed" their fingerprints to follow the same order as that in a data file or stream, and the topology and channel conditions are varied for a deduplication scheme to be network-aware in a mobile network. Network aware chunker consist of network-aware chunk size selection, Quicksync only stores one copy of all its chunks including real contents , all virtual chunks under different chunking strategies and the metadata. Metadata mainly consists of block list including all hashes of chunks and a vblock list including all hashes of virtual chunks. Vblock list includes hashes of virtual chunk, offset and length. If a chunk is virtual one, server fetches corresponding content based on offset and length of the chunk. The server forms the file according to its metadata, by using the chunking strategies the file splits and generate a metadata under various strategies. Quicksync does not transfer contents between two clients directly. A file split into two chunks and uploaded to server and server takes other strategies to get three virtual chunks that point to real contents. The content is fetched from the storage based on its pointer, when the server needs to update or send the virtual chunks. Redundancy eliminator is designed to eliminate the redundant sync traffic. Which is used to identify the input for encoding using the map of the new to the old version and the delta encoding will not provide any benefit if the two versions for encoding is not similar, but only involves additional computation overhead. Redundancy eliminator consists of sketch-based mapping and buffering uncompleted chunks, sketch-based mapping is the method in which the hashes of the new chunks are compared with those of the original file to identify the unchanged chunks that do not need to be updated and we leverage a technique named sketch to estimate the similarity of chunks in the two versions, for the chunks without a hash match in the original version. In buffering uncompleted chunks there exists two in-memory queues which was processed by the network-aware chunker to buffer the incomplete chunks. The chunks are stored temporarily using uploading queue which will wait to be uploaded in the network communication, with each chunk recorded with three parts: data content, hash value, and the sketch of it. From the network-aware chunker the new chunks are pushed into uploading queue and popped up if they are been completely uploaded. Therefore between a new chunk and a chunk in the uploading queue a map has been built.

4.2 Batched Syncer

The main factors that decrease the bandwidth utilization are the chunk sequential acknowledgement from the application layer and the TCP slow start for the sync of multiple small chunks.

Batched syncer has two methods Batched Transmission and Reusing existing network connections. In batched transmission the app-layer acknowledgment to maintain the chunk state is called as cloud storage services leverage. A connection interruption is one of a benefit to a client who needs to upload the unacknowledged chunks to resume the sync. To reduce the acknowledgement overhead small chunks into large chunks are bundled in the dropbox. Although this helps to improve the synch throughput, if the bundled one is not acknowledged all small chunks are retransmitted from the dropbox client, when there is a broken connection.

The app-layer acknowledgement is deferred at the end of the sync process and using connection interruption the unacknowledged chunks are actively checked. Due to multiple acknowledgements the overhead is reduced for different chunks and the acknowledgement between two chunk transmissions is avoided by the idle waiting process. After the capturing of a network exception by client, the check will be initiated which is caused by the process shut down or the connection loss at a local side, and the interruption in the network that cannot be easily detected by the local devices is another cause for the failure of the sync process.

The Reusing existing network connections is second technique used rather than making new ones in storing files. To make a new network connection for each chunk this method is used which is easy and natural, the new connections are created which extends the period in the slow start state especially in small chunks. The storage connection to transfer multiple chunks is reused by the batched syncer, in which the duplicate TCP/SSL handshakes are avoided. Persistent notification flows for capturing changes are maintained by cloud storage services.

5 System Implementation

It is unable to implement the techniques directly with the released dropbox software, unless the client and server of dropbox are totally a closed source. To synchronize the data with the dropbox server the user program is provided with API's by dropbox which is different from the client software, the API's are restful and operate at the full file level. A proxy is leveraged in Amazon EC2 which is close to the server dropbox to emulate the control server behavior. The map of a file is maintained to the chunk list and the hash and the sketch of chunks are calculated by the virtual chunks generated by the proxy. User data are first uploaded to the proxy and then proxy updates the metadata in the database and in the dropbox server the data is stored in the process of sync.

6 Performance Evaluation

The improvement of using a network aware chunker is investigated and to reduce the sync traffic effectively using redundancy eliminator is evaluated here.

6.1 Impact of the Network-Aware Chunker

The sync speed, defined as the ratio of the original data size to the total sync time, and the average CPU usage of both the client and server.

6.2 Impact of the Redundancy Eliminator

By using redundancy eliminator the sync traffic reduction is evaluated by setting a average chunk size to 1MB to exclude the impact of adaptive chunking. When TUO position is close to 1 it indicates that the modified content of the server is synchronized by some particular implementation. The TUO results for delete operation when it is close to 0 as the client does not need to upload the delta beside performing the delta encoding.

6.3 Impact of the Batched Syncer

Cloud storage services suffer low BUE when the average of chunk size is set to 1 MB to exclude the impact of adaptive chunking, to evaluate the improvement of various network conditions through network-aware chunker. When RTT is very low the unaggressive chunking strategy with low computation is selected by the client to split the file. In a poor network condition a small chunk size is selected to eliminate more redundancy and to reduce the total sync time. Chunking strategy which is more aggressive is applied to reduce the redundant data when Quicksync increases by the CPU usage both client and the server under the increasing condition of RTT.

Conclusion

Despite all this processes, mobile cloud storage services fail to efficiently synchronize data in certain circumstances. To address the inefficiency issues, Quicksync is proposed to support the sync operation with dropbox and seafile. By evaluations made the Quicksync can effectively save the sync time and reduce the traffic overhead.

References

[1] Y. Cui, Z. Lai, and N. Dai, "A first look at mobile cloud storage serv- ices: Architecture, experimentation and challenge,"[Online]Available: http://www.4over6.edu.cn/others/technical_report.pdf

[2] I. Drago "Benchmarking personal cloud storage," in Proc. Conf. Internet Meas. Conf., 2013, pp. 205–212.