Name: Ting-Yu Lan
AndrewID: tingyula

15-440 Distributed System
Project 2: File-Caching Proxy

Protocol between proxy and Server
This is a “check on use” protocol.

For each open RPC call, my proxy would first call server to get server’s all information expect file contents. Information includes, server’s file version, file length, the file exist status and if it is a directory or not. Then, proxy would check open permission, if any invalid scenario happens, proxy would return Errors.
Then, Proxy would check whether proxy got latest version. If proxy got latest version, and it is a read-only request, simply direct user to exists file. If proxy got latest version, but it is a write request, proxy would find latest version file locally, and copy a latest version for writer to use privately.
As to the scenario proxy does not have latest version, it would send another request to server for file content data.

Proxy would record whether a file is modified during it open and close period. If yes, when client close() that file, my proxy would push whole file to server. Then, server will update that file and file version. The version on server side is a monotonic increasing integer. My proxy use a hashmap to record each file’s version.

Consistency Model seen by the client
This project use open-close granularity for files. As a result, while client get a file from server/proxy, it would see that specific file content all the way till it close that file. It can only see up-to-data version (if some writer update that file) next time it open( ) that file. As to server side, last write would be the version being keep on server side. Also, that would be the latest version a proxy get for a request.

LRU replacement
LRUCache implement LRU cache replacement policy. It maintain a multi-thread safe CopyOnWriteArrayList. This list record reference sequence. When a file first reference, it would be added to this list. If a file being reference again, it would first delete is from this list, then add it to the end of the list. This sequence contains a self-defined CacheFile class. Within CacheFile, it has file path, length, and users number. While evicting files, I check users number. If it is equals to zero means no one is opening it now, so I can evict that file. If it is not equals to zero, I would leave that file in the sequence.

LRUCache also maintain cache size, it would keep updating available cache size whenever a file is added, delete, or its length is changing. While detecting available cache size is lower than zero, it would automatically call Evict( ) function to pick victims and remove them from cache.

Name: Ting-Yu Lan
AndrewID: tingyula

Cache Freshness
A writer private copy on proxy would be delete when it is close.
When proxy get a new version of file, it would go through all files it has. If there is a older version of that file, and no clients is using it, it would remove that file from cache to release available cache space, and ensure cache freshness. This allows all files in cache are those having chances to be use again in the future.

Cache Freshness helps proxy to have better performance. Since it can remove those would never be used in the future, it release more space for usable files. If no cache freshness design, next evict victim may be other usable file that being reference earlier, but latest version.

Data Transfer between server and proxy
In order to transfer all information in one RPC call, I create a ServerData. ServerData contains contents, version, file length, and errno. With this data type, I can reduce many RPC calls to reduce networking delay.

Handle sub-directory
My proxy save files in same directory as server does, expect saving it under cache root rather than server root. While getting new files from server, it is possible there is no such directory in cache. Thus, proxy would go through a loop of mkdirParents to construct that sub-directory for cache.

File Version on proxy
I use *ver *[version number]* to save different version of same read file in cache. *verw *[file descriptor]* to save private writer's copy.