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 the server to get all the server's information except file contents. Information includes the server's file version, file length, the file's existing status, and whether it is a directory or not. Then, the proxy would check open permission, if any invalid scenario happens, the proxy would return errors.

Then, the proxy would check whether the proxy already has the latest version. If the proxy has the latest version, and it is a read-only request, simply direct the user to the existing file. If the proxy has the latest version with a write request, the proxy would find the latest version file locally and copy the latest version for the writer to use privately.
As to the scenario that the proxy does not have the latest version locally, it would send another request to the server for getting the latest file content data.

The proxy would record whether a file is modified during its open and close period. If the file is modified, the proxy would push the whole file to the server when the client close() the file. Then, the server will update that file and file version. The version on the server-side is a monotonic increasing integer. My proxy uses a hashmap to record each file's version.

Consistency Model seen by the client
This project uses open-close granularity for files. As a result, while the client gets a file from the server/proxy, it would see a specific file content version all the way till it closes that file. If someone else updates the file, the client can only see that update next time it open() the file. As to the server-side, the last write would be the version being kept on the server-side. In other words, it would be the latest version a proxy requesting from the server.

LRU replacement
LRUCache implements an LRU cache replacement policy. It maintains a multi-thread safe CopyOnWriteArrayList. This sequence contains a self-defined CacheFile class. Within CacheFile, it has file path, length, and users number. My LRU use "close" to define which files are being used most recently.

I have a list recording a reference sequence. A file is recorded to the reference sequence when it is referenced for the very first time. The list also maintains how many users are opening files. If a file is being opened, it would not be evicted. Moreover, since I am using "close" to define the evict sequence, I move a file to the end of the list when a close( ) is called for that file.

LRUCache also maintains cache size, it would keep updating available cache size whenever a file is added, delete, or its length is changed. When the proxy detects available cache size is lower than zero, it would automatically call Evict( ) function to pick victims and remove them from the cache.

Name: Ting-Yu Lan
AndrewID: tingyula

Evict( ) function simply goes through the list in order. It has to make sure that it only evicts those files not being used by any users. It continuing evicting files until the available cache size is larger than zero.

Cache Freshness
A writer's private copy on a proxy would be deleted when it closes the file.
When a proxy gets a new version of a file, it would go through all files it has locally. If there is an older version of that file and no client is using it, it would remove that file from the cache to release available cache space to ensure cache freshness. This allows all files in the cache are those having the highest chance to be used again in the near future.

Cache Freshness helps proxy to have better performance. Since it can remove those that would never be used again in the future, it releases more space for usable files. If there is no cache freshness design, the next evict victim may be a usable file. Though the usable file is not the one being referenced most recently, it is the 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 saves files in the same directory as the server does, except saving it under its cache root rather than the server root. While getting new files from the server, it is possible that there is no such directory in the cache. Thus, the proxy would go through a loop of mkdirParents to construct that sub-directory for the cache.

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