1 Overview

This document is a record of data structure design of project 2.

There are three major components in the project: proxy, server, and cache. Proxy serves as client, and cache records the LRU of recently accessed files.

If a file is read only, it has two copies: a cache file that is inside the cache folder, and a server file that is inside the server folder. Otherwise, it has three copies: a cache file, a server file, and a temporary "local" file that is also stored in the cache folder and goes along with the client.

For each read/write update in this project, we read in a byte array with chunk size no more than 200000 bytes, and repeat until all content are read/written to avoid huge memory usage.

2 cache file

Cache file is the elements inside cache. For each cache file, it has 5 parameters: file: the file that is stored in linked list; path: the path of the file; isOpen: number of times the file is referenced; 0 if closed; version: version of the current file; option: open option of the file.

3 LRU replacement

Cache is implemented as a linked list with element type CacheFile, and is ordered from the most recently touched file to the least recently touched file. There are three functions for modifying the cache:

addCache: adding a new cache file into a linked list, it first check if there already exists a cachefile with the same path name; if yes, increment that cachefile's refernce and exit; then check if we need eviction, and keeps evicting the least recently touched closed files until there are enough space to put in the new item; finally insert the item at head.

updateCache: update the LRU for an item already in cache, it first finds the index of that item in linked list, and then delete it from linked list and insert again to the head. It also has another parameter that records the change of the file's byte length before and after the modification on proxy, and updates the cache's size, and requests evictions if needed.

deleteCache: delete an item from the cache.

4 consistency

We use check on use and open-close semantics. When a client opens a file, they will get a local copy of the most up-to-date file, and operate on this local copy only.

In proxy's open, we get the path of the file and analyze if cache file exists already. If the cache is in the cache, we compare if cached file's version is the same as the server's version: if the same, we only update the LRU and reference for the cache file; else, we update the cache file's content and version number with the content on server file to keep the cache storing fresh documents, and update the LRU and reference for the cache file. If it is not in the cache, we fetch the file from server, create a cache file, and write the cache file's content, update the version number, set the reference to 1. We then add this cache file to cache linked list.

In server, there are two functions corresponding to different operations: when the file does not exist on cache, the open function is called. We do error checking on file existence and open option, and if no error send the file content to proxy, and initialize version number as 1; when the file exists on cache, the update function is called. We first send the server file's version number to proxy, and if the version numbers from the two sides differ, we send the server's content to cache.

We also lock the cache on proxy side with synchronized whenever we need to do any updates on cache, and lock the version dictionary on server side when we are updating the versions.

This helps reducing unnecessary updates between proxy and server and also a consistency between different clients opening files. When there are multiple writes, each write will only operate on their local copy, and therefore not interfering with each other. When they close, the lock will ensure that only one version is being updated each time.

5 cache freshness

We keep the cache fresh by keeping a version number of server side and cache side. Version is kept in server as a hashmap between path names and version number integers.

In proxy's close, we first remove the local copy, if exists, from the cache linked list, and decrement the cache file's reference. Then, for non-read only files, we update the content of local file to server file and cache file. Finally, we close the random access files. We write the local file's content on server file, and update version number by 1.

but also helps my system to remain fresh for both the file. When a client open the file, they will get the most up-to-date copy of the file, and when they close it, the difference is immediately reflected on both the cache file and the server file.

6 protocol between proxy and server

Remote file is the protocol between server and client. It has a byte array that carries the bytes read / written from one side to the other side, and a field for recording any possible errors. When we want to write something from one side to the other, we wrap the data read from one content in a byte array, and send it to the other side. The other side unpackage the data and write it on its file.

7 proxy - read, write, unlink, lseek

In write, all files are non-read only. Therefore, we first get the random access files and write the content on them, and then update cache LRU for both the local file and the cache file. In read, we first fetch out the random access file from dictionary, and then read with the given parameters. We then check if the file is read only by checking it with the read only hashmaps, and if it is read only, we only update the LRU for the cache file; else we update the LRU for both the local file and the cache file.