Project Two Design Report

Design and Structure

'path', i.e. original path (e.g. huge.txt) is a key for almost all maps in my implementation of Proxy and Server. Having a 'path', we can map it into a cache master copy among many version cache of this cache. We also

1. Compare and update version

Both Proxy and Server keep a map mapping path to its version.

When the client **reads** a 'path', the server will compare the master cache copy mapped from 'path' with the server's version. If the version is the same, directly read content in the master cache copy. If not, delete old master cache copy and fetch the latest version from the server.

When a client **writes** a 'path', the server will also compare the master cache copy mapped from 'path' with the server's version. **If matched**, directly copy local old master cache copy to a new unique path and write on it, after finish writing, try to delete old master cache copy and change the map_path_master to map 'path' to this new unique path and insert it into LRU queue. From then on, this new unique path is a master cache copy corresponding to this 'path'. **If not matched**, just write on a new file and establish its map to 'path'.
The server only updates its file version when proxy finishes a write.
Proxy changes its file version from fetching the version from Sever.

In conclusion, the writer will write on its unique copy and a copy will become master cache after writing and push to the server. The reader will directly read on the master cache copy.

2. When to delete old master cache copy?

When we want to delete an old file. We should take whether some readers are reading on it into consideration. In proxy, there is a map to map 'path' to 'readcount', When the reader start reading, "readcount++', when finish reading, readcount--".After writers finish writing a file that once existed in the proxy(i.e. finish writing on its copy), check the readcount of the old file. If readcount == 0, delete it. If there is a reader reading on it, let that reader deletes it after finishing reading.

3. How to implement LRU replacement

My LRU is a class that can automatically evict cache and deletes file after a new cache pushed in it and the current size exceed the set capacity. After every 'put' operation at reading's and write's 'close()', if exceed the set capacity, LRU class will choose a victim with 'readcount == 0', unlink all its map and delete its file. Because my write implementation is writing on a copy of an existed file, it may cause exceeding the capacity during writing. Therefore, My LRU also implements forceevict() method to pre-evict a victim before writing

on a copy.
The way to get an LRU order is to use a LinkedHashMap() object with accessOrder=true.

4. How to ensure concurrency?

At proxy, there is a static object ‘cachelock’ as a global lock. Every time getting access to ‘global variable’ like the map, LRUcache, etc., proxy will lock this cache lock to ensure concurrency.

5. How to transfer data between proxy and server?

Before read and write, the proxy will compare the version. If the version is out of date, proxy will fetch the latest file from the server in chunk. After every write, proxy will write in chunk to update the corresponding server’s file.

03/01/2020
Zihao ZHOU