Project 2 Design Document

Peizhao Li peizhaol@andrew.cmu.edu

1. Overview Diagram: see the attachment image on the last page.

The program has a very good modularized and object-oriented design. The attached image on the last page demonstrated a simplified abstraction of the project.

2. Design Specifications

2.1. Protocol between proxy and server: ReplyFileInfo class

Data Abstraction	FileInfo.java	ReplyFileInfo.java	CachedFileInfo.java
What data does it contain and what is it used for?	Used to store RandomAccessFile and related information such as the file's access_mode	Used to send a reply from Server which contains remote file information such as boolean is_existed, boolean is_dir, int remote_version_num, etc	Used to store cached file entries and for LRU maintenance. It contains reference_count to avoid potential concurrency issues, file_size to do evict()
Where is it used?	Proxy	Server -> Proxy	Cache

2.2. Consistency Model: Check-on-Use

2.2.1. open()

On each access, the Proxy first acquires a ReplyFileInfo from Server about the remote file's information including is_existed, is_dir, remote_version_num, etc. Then the Proxy checks the local cache to decide whether to fetch the file from a remote Server (cache miss) or just use the local file (cache hit). A series of operations on cache are also encapsulated here in method deal(), such as create_write_copy() on "rw" modes, add_reference_count(), increase cache_size, etc.

2.2.2. close()

On close, it sends the updated file back to the Server if it is a "wr" access mode. Furthermore, it marks the file as MRU by appending or moving (existing) CachedFileInfo entry to the tail of the LinkedList.

Throughout the process, the reference_count is implemented to avoid eviction or unlink of currently used files. When a write file is closed, it is uploaded to Server and saved in cache with the latest version number.

2.3. LRU Cache Implementation:

2.3.1. Basic Logic

LRU is maintained by a LinkedList. On both open and close, the CachedFileInfo entry is appended (new entry) or moved (existing entry) to the tail as the MRU file.

Moreover, delete_old_versions() and evict() are implemented to ensure the cache is not occupied by stale and useless files. The delete_old_versions() method is prioritized to remove the files that

would not be used again. These files always have the latest version copy in cache but were not evicted before due to previously being read.

2.3.2. Structures used to ensure LRU

Data structures	LinkedList cache_line	ConcurrentHashMap path_file_map
Elaborations	Maintain a list of CachedFileInfo entries which contains reference count, file_size, etc.	Maintain a hashmap which contains path-CachedFileInfo mapping to help indexing and build the cache abstraction.

2.4. Way to ensure Cache freshness

Cache is guaranteed to be fresh on each access. This is achieved by comparing the local cache version number of the file and remote version number of the same file. If the cached content is stale, it fetches the latest file from Server and updates a series of local cache information as described above.

2.5. Concurrency related implementations

Where	Proxy/ Cache	Server
Synchronizations	Object lock Object cache_lock Synchronized methods ConcurrentHashMap fd_file_map	ConcurrentHashMap path_version_map ConcurrentHashMap path_lock_map

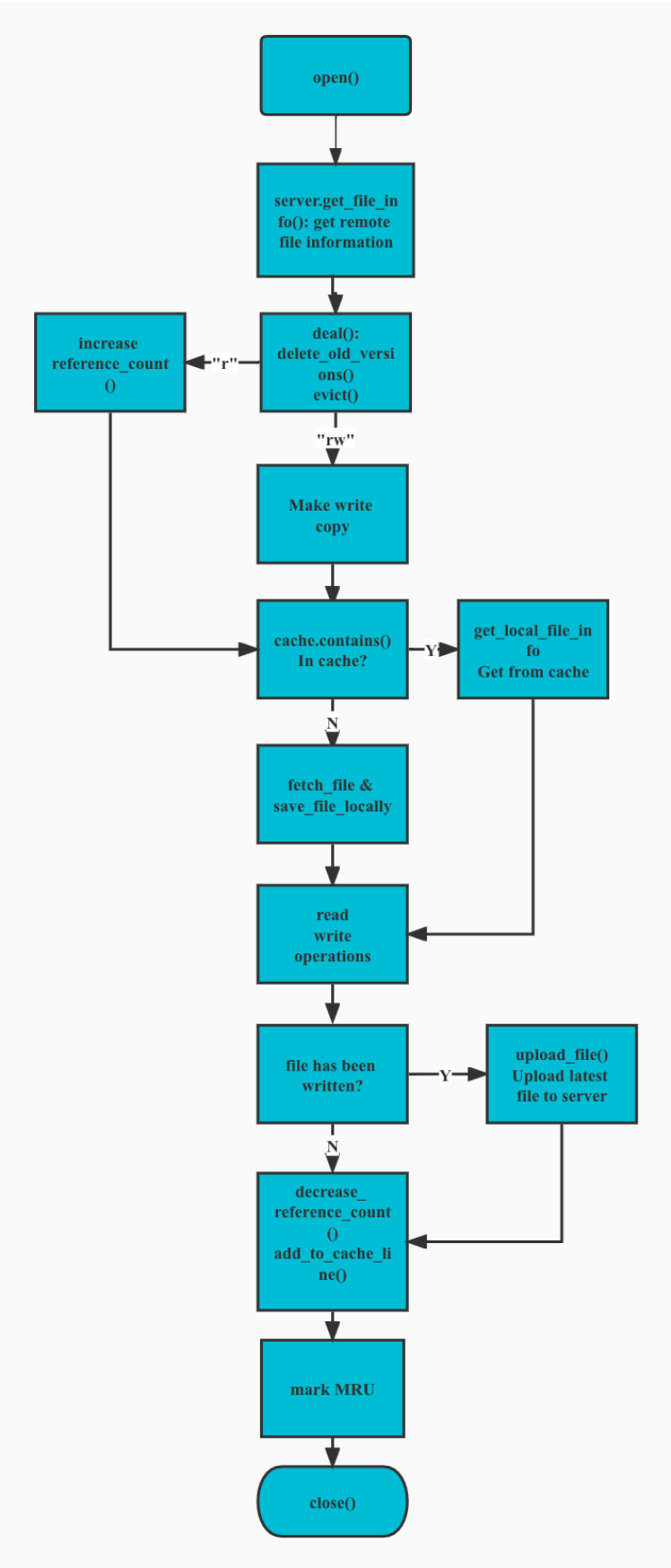
There are 2 major aspects of concurrency.

2.5.1. Handle concurrent clients.

The Proxy uses Object lock to ensure file descriptors are assigned uniquely to each opened file. It uses Object cache_lock with the keyword synchronized to ensure multiple accesses to the cache's LinkedList or HashMap does not create a race condition. Moreover, to allow concurrent read/write operations, it creates a unique copy for each write operation and names the write copy with its file descriptor. Only in this way can the file have open-close semantics for read/write at the same file.

2.5.2. Handle concurrent proxies.

The server assigned a similar Object lock for each file on the remote Server to avoid races in upload_file() from Proxy to Server and get_file() from Server to Proxy.


open()
server.get_file_info(): get remote file information
deal(): delete_old_versions() evict()
"r"
increase reference_count()
"rw"
Make write copy
cache.contains() In cache?
Y
get_local_file_info Get from cache
N
fetch_file & save_file_locally
read write operations
file has been written?
Y
upload_file() Upload latest file to server
N
decrease_reference_count() add_to_cache_line()
mark MRU
close()