CS 8803 Intro to OS Project 3 ReadMe

Yan Cai ID: ycai87

Part1:

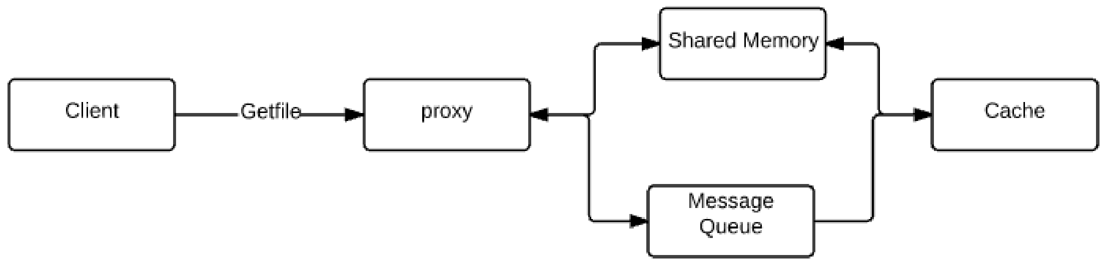
Proxy only:

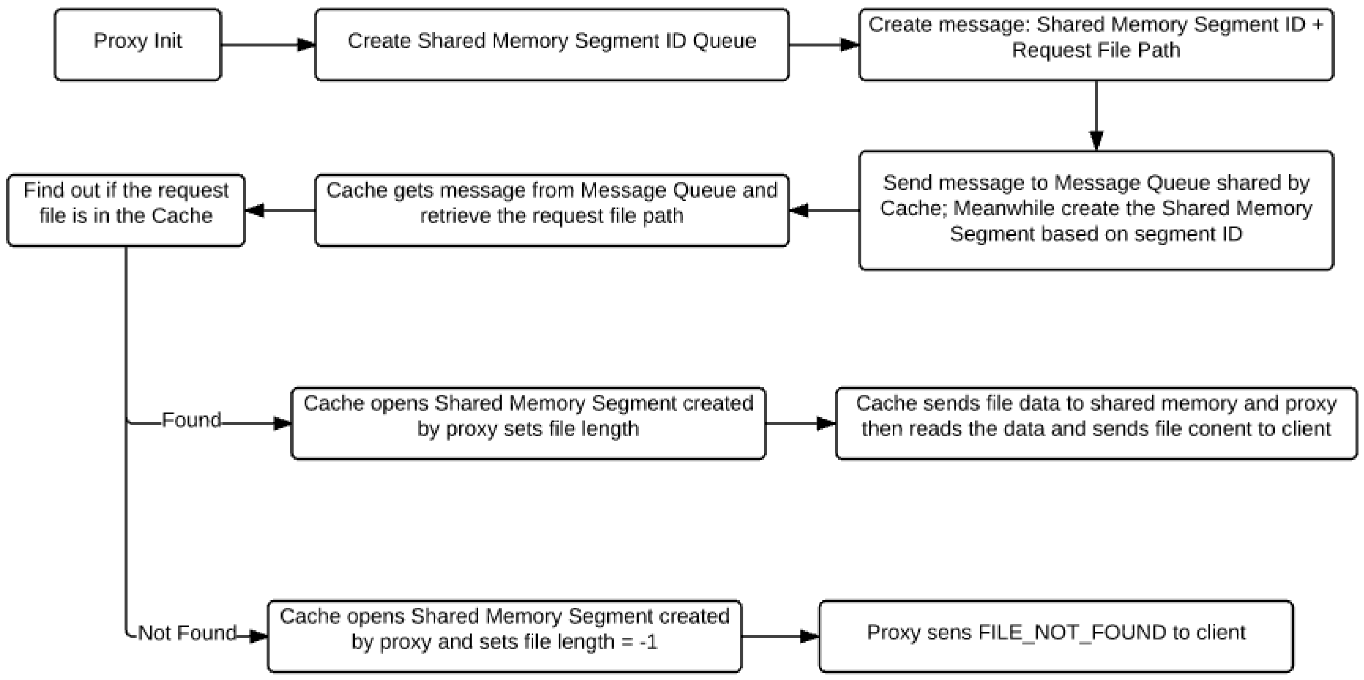
Create a memory chunk and store all the data from libcurl operation. If the file is found with the link in the server, the proxy then sends the data from the memory chunk to the client. The memory chunk works as a temporary place for content transfer.

Part 2:

Proxy and Cache:

The following is the design flow graph:




Proxy Init
Create Shared Memory Segment ID Queue
Create message: Shared Memory Segment ID + Request File Path
Send message to Message Queue shared by Cache; Meanwhile create the Shared Memory Segment based on segment ID
Cache gets message from Message Queue and retrieve the request file path
Find out if the request file is in the Cache
Found
Cache opens Shared Memory Segment created by proxy sets file length
Cache sends file data to shared memory and proxy then reads the data and sends file conent to client
Not Found
Cache opens Shared Memory Segment created by proxy and sets file length = -1
Proxy sens FILE_NOT_FOUND to client

Flow explanation:

1. Proxy maintains the shared memory segment ID queue which is determined by –n : number of segments. Initially, proxy creates the memory segment ID strings and puts in the queue. Every time proxy receives a getfile request from the client, proxy pops the queue and then gets a shared segment ID.
2. Proxy then creates a shared message: segment ID + getfile request file path. Meanwhile, proxy opens a shared memory segment based on segment ID. Proxy then sends the message to the message queue shared by cache.
3. Cache also maintains a message queue stack to itself for reference. Cache receives the message from the shared message queue and puts the message queue data to its own stack.
4. Cache gets the requested file path from the message data and tries to find if the file is in the cache by using simplecache_get(msg->path);
5. If the file is in the cache, cache will open the shared memory and sends the file data in the cache to shared memory segment. If the file is not in the cache, cache will set the file length = -1.
6. Proxy will read the file length in the shared memory, if the file length is > -1, it will send the shared memory data to client, otherwise, it will send FILE_NOT_FOUND to client.
7. Proxy will clean up the shared memory and enqueue the used segment ID and start over for the next incoming request.

Synchronization constructs:

1. For Proxy worker threads: Use mutex and condition variables to maintain the segment ID queue.
2. For Cache worker threads: Use mutex and condition variables to maintain the message queue data stack.
3. Message queue shared between proxy and cache: the message queue API is automatically synchronized by kernel. No extra constructs are created.
4. Shared memory segment shared between proxy and cache. Create shared memory segment data type, which includes mutex and condition variables whose attributes are PTHREAD_PROCESS_SHARED. Proxy and cache will read/write the shared memory segments through synchronization constructs.

Test results:

3 Categories of tests were conducted:

Here N >= 1.

1. Cache N threads; Proxy N threads but 1 memory segment, client N threads: PASS.
2. Cache N threads; Proxy 1 thread but N memory segments; client N threads: PASS
3. Cache N threads; Proxy N threads and N memory segments; client N threads: Unstable.

In #3 situation, when N gets larger, the client will get stuck. Possibly because of the deadlock created by multiple threads and multiple segments in the proxy. A possible cause could be when multiple proxy worker threads get same memory segment ID thus same shared memory yet different file requests, dead locks happen. I could not figure out the solution given the time.

Observations:

1. In message queue send and receive, the size of the message needs not to be the same as the message queue data, because it may cause error to send or receive messages. I set it 1024 here.
   mq_receive(m_queue, (char *) &data, **1024**, NULL);

References:

1. https://curl.haxx.se/libcurl/c/getinmemory.html. It gives a good reference on how to create a memory chunk and stores the http request data.
2. https://github.com/newtonrd/CS8803-Project3/tree/master/proxy-cache. This one creates a lot of extra function calls for shared memory steque structure, which are not necessary. However, it does not work with N memory segments and N proxy threads.