Initial design

Summary:

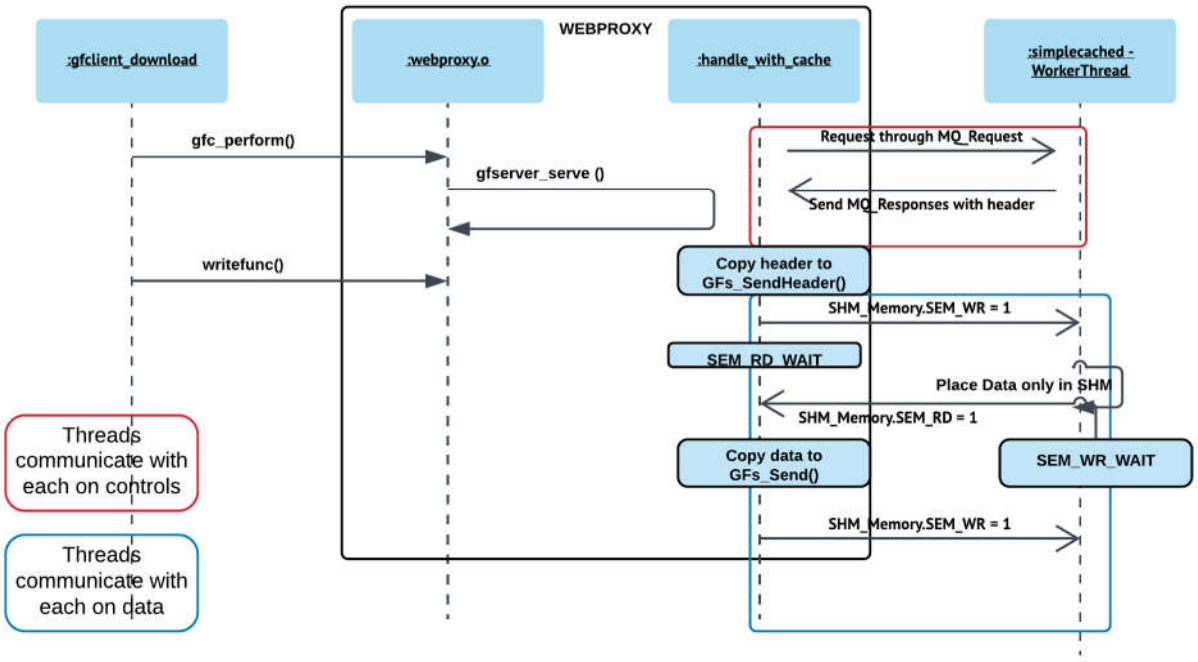
Boss Worker thread pattern with each service to communicate.

2 Message queues for workers to communicate among themselves about a given request.

WHY: This design, will make different threads (simplecahed and webproxy_handler) to handle request and response independently, improves efficiency.

On two message queues

1. One for, WebProxy to request simplecached.
2. Second for, SimpleCached to respond WebProxy that the data is ready to be transferred or file not found.



Here I used

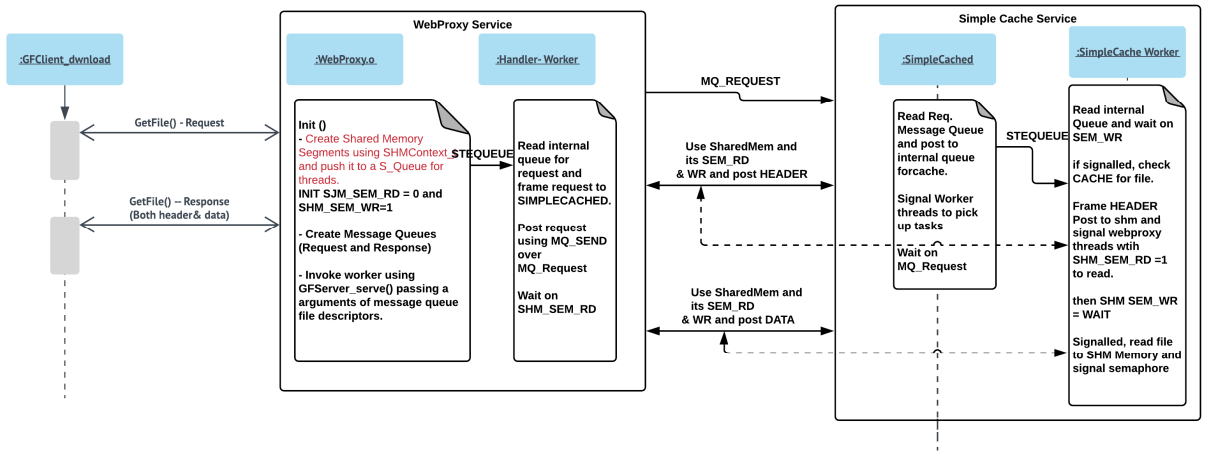
1. MQ_Request for sending requests from Webproxy Service to Simplecached Service.
2. MQ_Response for sending a response from Simplecached Service to Webproxy service.
    a. Why: Message queue synchronization is taken care my Linux kernel, so preferred initially to use more message queues.
    b. Easy to synchronize across threads in multiple processes using message queue for posting found, file_not_found status back to GFClient_download.
    c.
3. I used SHM_Context to post file data only.

Second design

Removed Message Queue RESPONSE from control channel communications.

Here, I used the shared memory data structure to fill all other context information needed between the proxy boss thread and worker threads, to avoid malloc.

But, I had other semaphore lock problems and redesigned with the 3rd approach below.



Final design

1. Single (Message QUEUE) Control Channel
2. HEADER and DATA are using shared memory with binary semaphores (1 for reading and 1 for write) as initial design.
3. Moved separated proxyContext used for communicating within (Boss & Workers) of WebProxy and SHMContext. It created allocation and free of nSegment of pointers during SEGINT.

Issues and how addressed

1. SEGSIZE < sizeof(SHM_CONTEXT_STRUCT_T) Issue

What caused:

SHMContext and WebProxyContext are merged and stored in SHMemory. The webproxy-workers reads the internal queue to get this SHM pointer and retrieve other webproxy specific parameters too (nsegSize for calculating databuffer start address inside allocated SHM).

Due this is too many parameters, my struct size was 4208, and in one of Bonnie's tests I noticed, 4096 is segsize.

How diagnosed: I added each service's arguments to be printed in the log while testing against Bonnie. Which helped me to see this value even I intended to see arguments to simulate locally.

How addressed: Separated WebProxy and SHMContext into two different data structure. Pass the WebProxyContext through steque and general parameters as thread's argument from the main thread.

2. mq_send fails with BAD_ADDRESS

What caused:

I allocated 6200 when MQ_OPEN using attributes. This issue never happened to me on my local testing with multiple threads of all two services and client.

How diagnosed:

Bonnie test case failed. I have added pretty much all the failure cases with a traces; this helped to diagnose quick. Then, to understand the issue, StackOverflow and manpages are used.

How addressed:

By changing the msg_size parameter from 6200 (macro) to sizeof the request what's being posted fixed.

3. mq_receive needed more than allocated MAX_MSG_SIZE

What caused:

I allocated 6200 when MQ_OPEN using attributes.

How diagnosed:

Bonnie test case failed. I have added pretty much all the failure cases with a trace; this helped to diagnose quick. Then, to understand the issue, StackOverflow and manpages are used.

How addressed:

In mq_receive I increased the msg_len as allocated size given while MQ_OPEN +1.

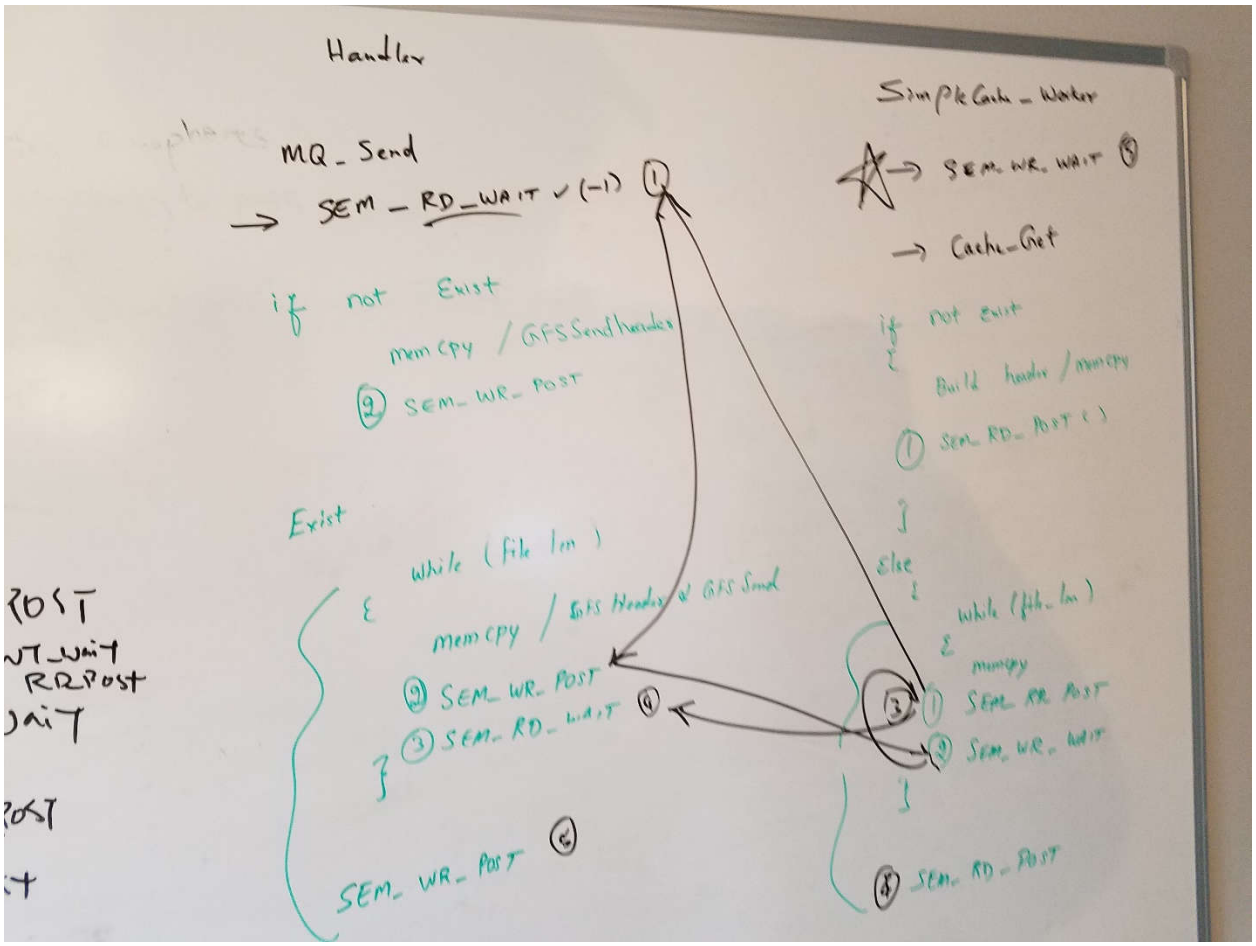
4. simplecached single threaded file handling failed with partial file not transferred by webproxy_hander

What caused:

If the simplecached service runs as a single thread, gfclient and webproxy as multi-threaded with multiple requests. Particularly with t=8 and r=8 and segsize=4096 the data transfer of file contents partially dropped.

How diagnosed:

Added traces, whiteboard the semaphore allocations and rewrote the shm_handling logic with simpler.



How addressed:

Rewrote the signaling part with simple if & else blocks. Tested with same arguments used by Bonnie locally first to create a test environment and then validated my fix and pushed to bonnie's approval.

5. Failure of Bonnie's "Cache Test with single threaded downloads" and "Cache Test with simultaneous multi-threaded downloads (mixed file sizes)" test cases

What caused:

In solving issues #4, I introduced a bug in the FILE_NOT_FOUND scenario. If I make a file_not_found locally and test I tested with one file request so made me overlook. While, running in Bonnie, existing and non-existing files are queue together, which exposed this bug. I had multiple SEM_WRITER_POST only in the file_not_found code flow.

How diagnosed:

Code reviewed again with the white boarded picture and specifically file_not_found scenario.I updated workload.txt and local,.txt to simulate the same locally to validate my fix since I had only three bonnie submissions left for the day.

How addressed:

Cleared the duplicate SEM_WRITE_POST inside webProxy_handler and completely cleared any specific signaling for the FILE_NOT_FOUND scenario. So, by doing this, if a file_not_found, simplecached will post the response and wait for next request from BOSS thread instead of the handler to respond back.

Generic problems:

Memory leaks: When I create a malloc call I also made sure to call free with that object the same time, which helped to have less problem with a memory leak.

Whiteboarding helped to understand the lock mismatch issue and other specific code flow.