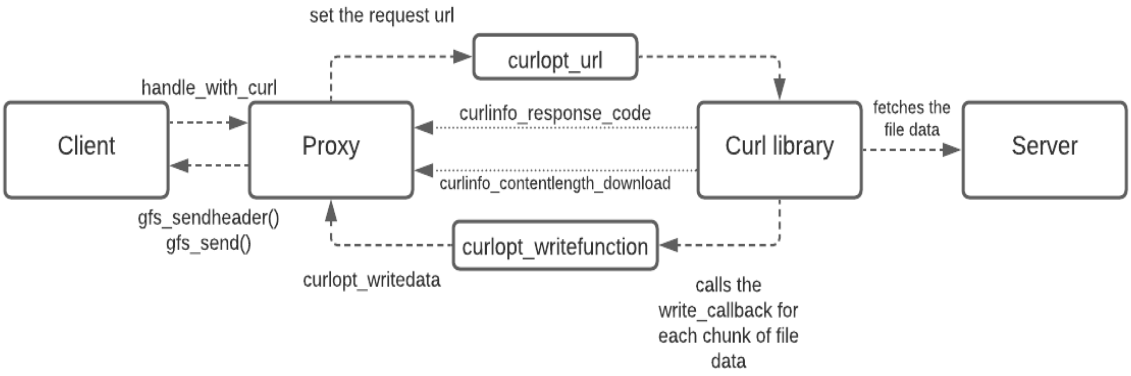
README

Introduction

This is GIOS Project III Spring 2024 Readme file from course CS6200. It has two sections.

1. Proxy server
2. Cache server

Part I: Proxy Server



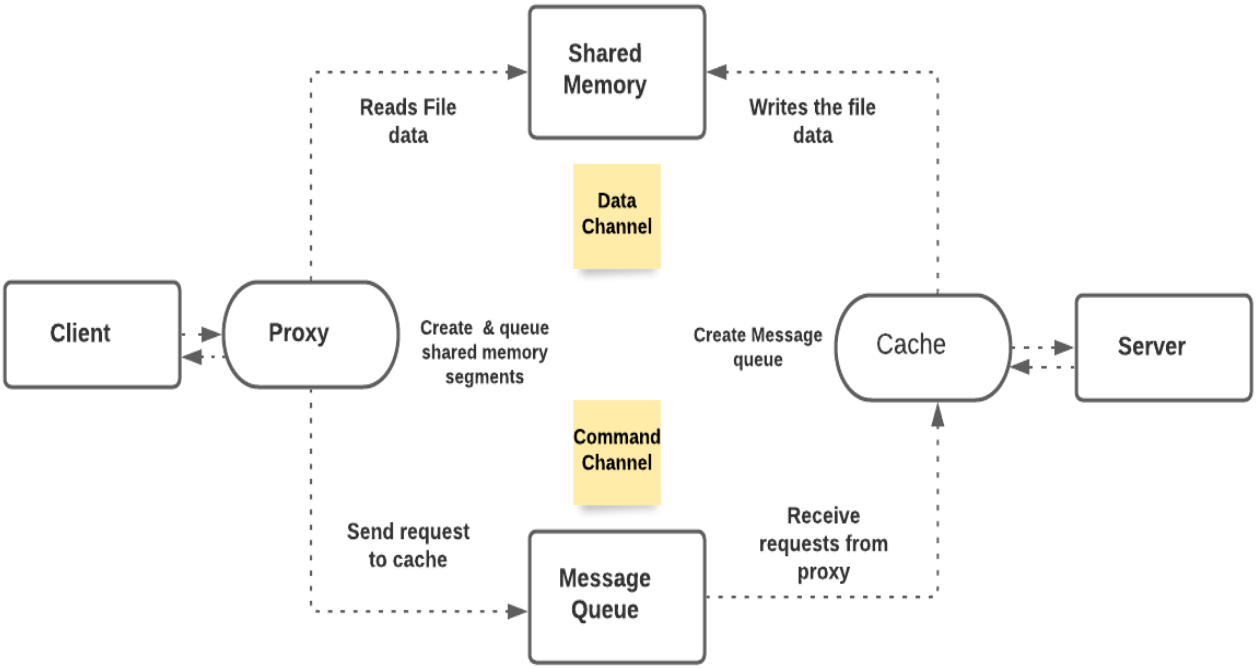
The multithreaded proxy server acts as a mediator, handling requests from clients to servers. It utilizes libcurl to query webservers and fetch files. The request URL is set using CURLOPT_URL. After a successful curl perform operation, the response code and file length are fetched using CURLINFO functions. The proxy then sends corresponding header to the client via gfs_sendheader based on the server's response code. For each data chunk read by libcurl, the registered CURLOPT_WRITEFUNCTION callback is invoked, filling the CURLOPT_WRITEDATA buffer with file content. This buffer is then sent to the client using gfs_send, repeating until all file content is delivered.

Testing:

1. Single proxy thread tested against varying client request volumes, providing appropriate responses per requested file.

2. Multiple proxy threads successfully handled diverse client request loads, downloading requested files.

Part II: Cache Server



Project Design :

Command channel : **POSIX message queue**

Data channel : **POSIX shared memory**

File transfer between client/server via proxy/cache was implemented using Inter-Process Communication (IPC). The multithreaded client sends requests to the multithreaded proxy, which passes them to the cache through a command channel. Cache

server receives the requests and serves the file contents back through Data channel.

Implementation

The proxy server first creates and resizes shared memory segments (shm_open, ftruncate) and maps them to its process space using mmap. As pointers from one process are not visible to another, all metadata and synchronization constructs need to be on shared memory. The shared memory data structure has a flexible array for file data along with metadata. Mapped shared memory pointers (cast to structs with distinct segment IDs) are placed in a steque, protected by a mutex and condition variable for thread synchronization.

Client requests are tagged with a shared memory pointer from the steque and sent to the cache via a message queue. Then the pointer is enqueued back to serve other requests. Request concurrency depends on available free segments; threads wait on the steque for free memory when segments are busy.

The cache uses a boss-worker model. The boss creates the message queue and workers. Workers retrieve requests from the multi-thread safe queue and process them. Three semaphores signal file status, file size, and read/send synchronization between cache and proxy.

Each request uses a shared memory segment during transfer. The cache maps to the segment, resizing it per file size. If the file is not found, the status is updated, and the proxy is notified via semaphore. Found files have their sizes updated, and data is read in chunks, memcpied to the shared buffer. The cache waits while the proxy sends data to the client, signaled to read the next chunk until all data is sent.

When the file is successfully transferred, the memory pointer is enqueued to be reused.

Testing:

1. Began with single-threaded client, proxy, and cache to test successful single request processing for GF_OK, GF_FILE_NOT_FOUND cases, and file downloads.
2. Implemented multithreading for proxy, client, and cache, testing with varied request volumes, thread counts, segment counts, and segment sizes.
3. Tested scenario where requests exceeded available segments, verifying dequeued and re-enqueued segments were successfully reused by proxy threads.

References:

“C++ - Sending Image (JPEG) through Socket in C Linux.” https://stackoverflow.com/questions/15445207/sending-image-jpeg-through-socket-in-c-linux.

“C Library Function - Memset() - Tutorialspoint.”

https://www.tutorialspoint.com/c_standard_library/c_function_memset.htm

Curl.haxx.se. 2020. *Libcurl Example - Ftpuploadresume.C*.

C++, S., Isaksson, J., Laagland, M. and Köhler, U., 2020. *Save Curl Content Result Into A String In C++*. Curl.haxx.se. 2020. *Libcurl Example - Getinmemory.C*.

“C - Differ between Header and Content of Http Server Response (Sockets).”

https://stackoverflow.com/questions/16243118/differ-between-header-and-content-of-http-server-response-sockets.

Users.pja.edu.pl. 2020. *POSIX.4 Message Queues*.

“POSIX Threads Programming.”

https://hpc-tutorials.llnl.gov/posix/#PassingArguments

Referred the below Github file to understand the process control flow

https://github.com/xericyang97/GIOS/tree/main/Project3

Photo from the below Github file control flow

https://github.com/ JianchengGuo/GIOS6200/ /tree/master/

Gist. 2020. *POSIX Shared Memory IPC Example (Shm_Open, Mmap), Working On Linux And Macos*.

Kohala.com. 2020.

GeeksforGeeks. 2020. *Flexible Array Members In A Structure In C - Geeksforgeeks*.