Gebze Technical University
Tugay Apaydın
1801042081

CSE344 – System Programming
Final Project - 2020

REPORT

Commands

Server Command:
./server -i InputFile -p PORT -p LogFile -s INITIAL-SIZE -x MAX-SIZE

Client Command:
./client -a "Any IP address-p PORT -s SOURCE -d DESTINATION

I started the homework by firstly reading the command line arguments. After that, the first thing that I have done was to build all required data structures (queue, stack, hash table (for caching), graph, and a different type of a linked list structure).

Firstly, I implemented a graph to my program by using adjacency list implementation. In order to do a bread first search and create a path, I needed stack and queue implementations. That's the only place that I used the stack structure.

Second, I implemented the queue structure in order to store available threads (as a thread pool) and use to find a path in bread first search for graphs.

While explaining the queue implementation, I can also talk about the data pool which refers to thread pool. I only stored the id numbers of the available threads in the queue. Once a thread becomes available, it pushes it's id number to the queue so that main process can take it as an id of an available thread. Of course this is a synchronized process since queue is a critical section. Both main thread and all other threads blocking the pool before modifying it. After main thread takes an available thread from the queue, it sets an "available"flag to "FALSE"and sends a signal to the thread. If thread reads this "available"flag as "FALSE", that means the main thread made him busy (main thread basically saying that you have a work to do). After that, thread reads the client socket file descriptor and calculates the path by using graph functions.

CACHE STRUCTURE IMPLEMENTATION

For each calculation of a path, threads first check the cache data structure which is implemented by using a Hash Table. I have chosen the Hash Table in order to

implement cache structure because on average it gives for inserting and searching O(1) time efficiency. Which means that, it is the best structure for caching since we are looking for a better searching and inserting efficiency.

After each calculation of an existing path, if thread cannot find the path in the cache; it firstly calculates it by using BFS algorithm, then inserts the calculated path to the cache. Cache structure is also a critical section and more then 1 process cannot access to the cache structure since it is synchronized. Caching process only synchronized between threads that calculates a path.

DATA POOL IMPLEMENTATION

I have explained that threads are stored in a queue which used as a thread pool. There is also a linked list structure which stores everything (threads, mutexes, all required data sources like graph, queue etc.) in it. Once the program has started, it creates a data pool with a node which stores initial size of threads in it. After each "enlarging"which is done by "resizer"thread, a new node with a new size is added to the end of the linked list. While creating a new node, all threads are made available to access to the common data structures by using pointers. Also, for an exit condition; destroy pool function frees all allocated resources and destroys everything to be ready for a shutdown.

RESIZER THREAD

By using the advantage off data pool implementation, resizer thread can basically enlarge the thread pool by using enlarge function. After each process, main thread lets resizer thread to check the queue. If the %75 of the threads in the pool are busy, it extends the pool to %25 more elements.
For each request, main thread should take a thread from the thread pool. That's why, thread pool is only critial section between main thread and resizer thread. All other threads can also access to the pool but they will access to the node that they are stored. On the other hand, main thread may try to check all nodes in order to find a thread. Because of that; for each checking and extending processes, there is a synchronization between main thread and resizer thread.

EXIT CONDITION

If SIGINT signal received by the server, it basically sets an exit flag to 1. After server saw that flag, it calls destroy pool function and shutdowns the server. All other threads can also see the exit flag by using data pool structure and destroy pool function also joins them before shutting down the system.

EXIT CONDITION

Client process only connects to the server and requests a path. After receiving a message from the server, it basically prints it to the terminal.