CSE344 System Programming Extra Project Report

August 15, 2021

Student: Furkan Sergen Dayioglu
Student No:121044015
Instructor: Erchan APTOULA

1 Overview

In this assignment, we were asked to implement a low cost but in efficient server client architecture.Where multiple client of X tries to connect to Y. And if Y is busy then it will use ServerZ's hardware/ thread And then if Z if busy then it will use other servers hardware/thread.

During this assignment we had one client code, and two server codes.

ClientX will read a matrix from csv file and upload it to server. Then server send the information about matrix if it is invertible or not.

Server Y will handle the connection of Client(s)X with its thread pool. If the thread pool of serverY is full then other other thread pool will connect to the Server Z send matrix to it and ServerZ send response to ServerY and it sends the response to the ClientX.

./clientX -i ID -a IPADRESS -p PORT -o DATAFILE

./serverY -p PORT -l POOL1 -m POOL2 -t SLEEP_DUR -o LOG_FILE

./serverZ -p PORT -m POOL -t SLEEP_DUR -o LOG _ FILE

Server Z will not be run by us explicitly but by ServerY

2 Synchronization Between Threads

Synchronization between threads are handled by mutexes and condition variables. As it is asked to implement, i implemented a wait queue. So i can store the socket file descriptor of the connected client. And since the implemented queue is not thread safe, i used a main_mutex to make it thread safe.

For Server Z part; it is the first part that i implemented because its calculation logic is similiar to ServerY pool1 threads, main_mutex and a condition varible were enough. When i enqueue a client socket i send signal to mutex to read it. According to video o found on youtube which is uploaded by Jacob Sorber, checking (dequeueing) the file descriptor in the head of queue is not enough, we must wait a condition and dequeue again. It supports protection aganist NULL file descriptor and also even if there are no new incoming connection, it dequeues and handle remaining connections.

For ServerY part, it is a little bit tricky. I used one mutex to make queue operations thread safe, but i used two condition varibles and two mutexes to check if the pool1 or pool2 is available. According to rules that specified in submission paper, we should check pool1 first. If there are no avaliable threads, then we should check pool2, if both pools are busy, we should go through the first available one. I dont think if i can manage that part, but i check if the pool1 is available, then send signal to pool1 thread to dequeue and element. Else i send a signal to pool2 thread to dequeue an element.

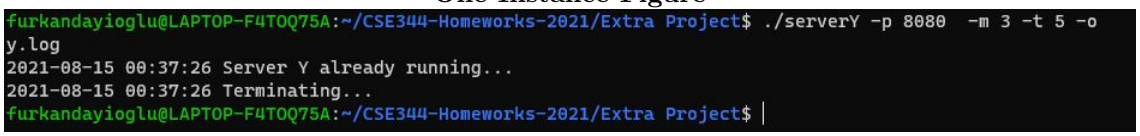
3 SIGINT situation

When termination signal come, it has been asked to running threads will finish their jobs gracefully, and free their resources and then join.Signal handlers change sig_int flag and threads checking it if SIGINT occured.

4 Avoiding Double Instantiation

In this part, i created a temp file under tmp folder and locked it.It is same for ServerZ too.

One Instance Figure



5 Thread pool

I created pool1 and pool2 for ServerY and created pool for ServerZ. They get threadPoolY
Z structure as parameters in order to determine what is their id, which port they are going to connect, sockets that they will read, their status.

6 Invertible

The criterias of being invertible matrix are being square matrix and determinant of it must be 0. I calculated determinant of the matrix according the code i studied from geeksforgeeks. Then set a response value. If determinant of matrix is zero, response is zero, else response is 1.

7 The question on the submission paper

The question that is asked why this paradigm is not so smart. In my opinion, it is not smart because serverY is both load balancer and server at same time. If serverY would be load balancer and the other serverZ's would be servers, it would be smarter. I say this because as we know there is a service that named as CloudFlare, which provides webservers aganist DDoS attacks. When a server is blocked, it flares another image of last working version of server, and keep running. In this way, serverY can do something similiar. Threads of Y checks other server according to their availability, and then delegate the connection to avaliable one. If there are no available then create one more instance.