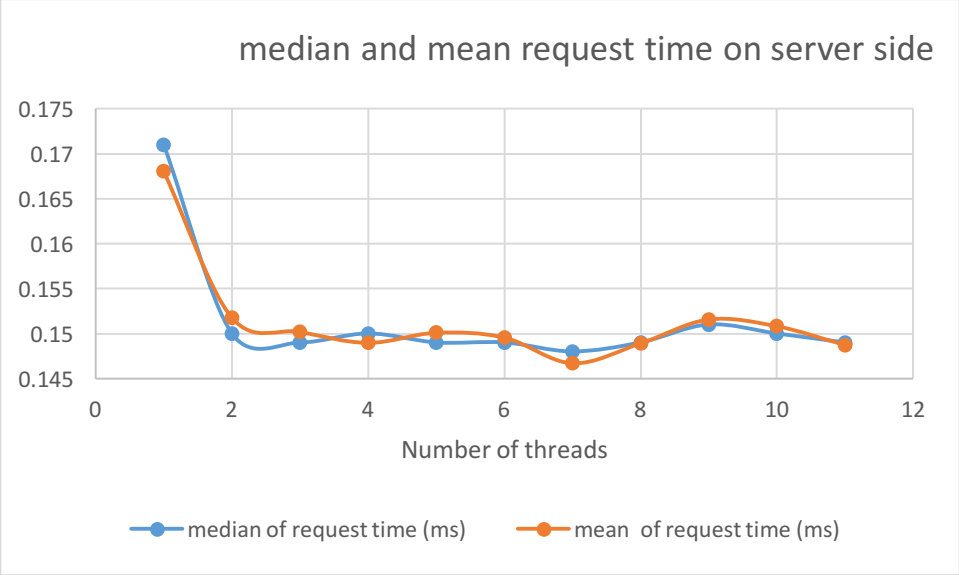
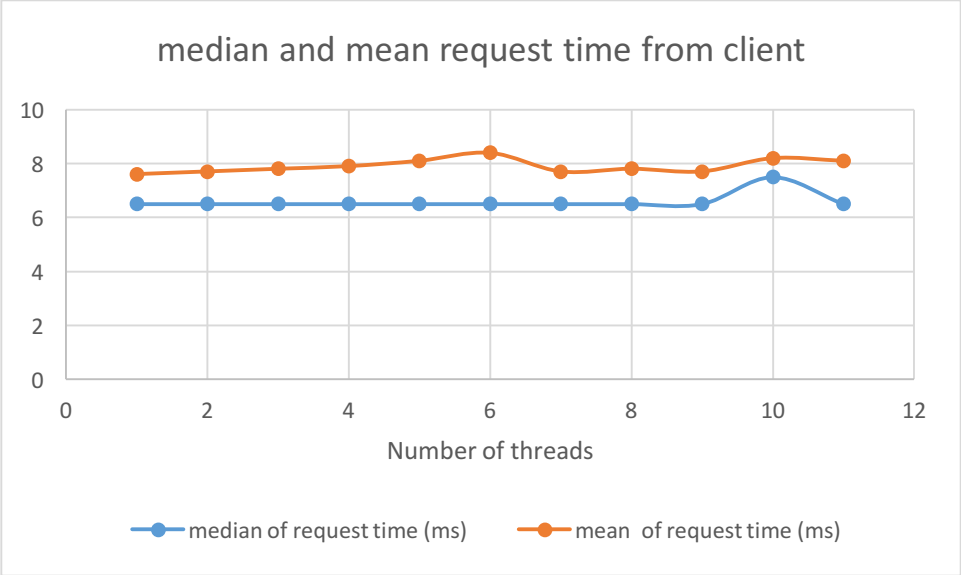
Optimization methods I used:

1. use several locks based on purpose instead of one global lock.
   Each lock is for a specific purpose. If use a single exclusive lock, then the waiting time should be very long for each block that needs the lock. Using several locks based on purpose can largely save time on waiting (reduce portion of sequential part).
   But for my implementation, I didn't see much improvement.
2. use conditional variable. At first, I used busy waiting. This method let all the threads running all the time, and running the threads occupies lots of computer resources. To save enough computer resources, I only wake up one thread when a connection comes.

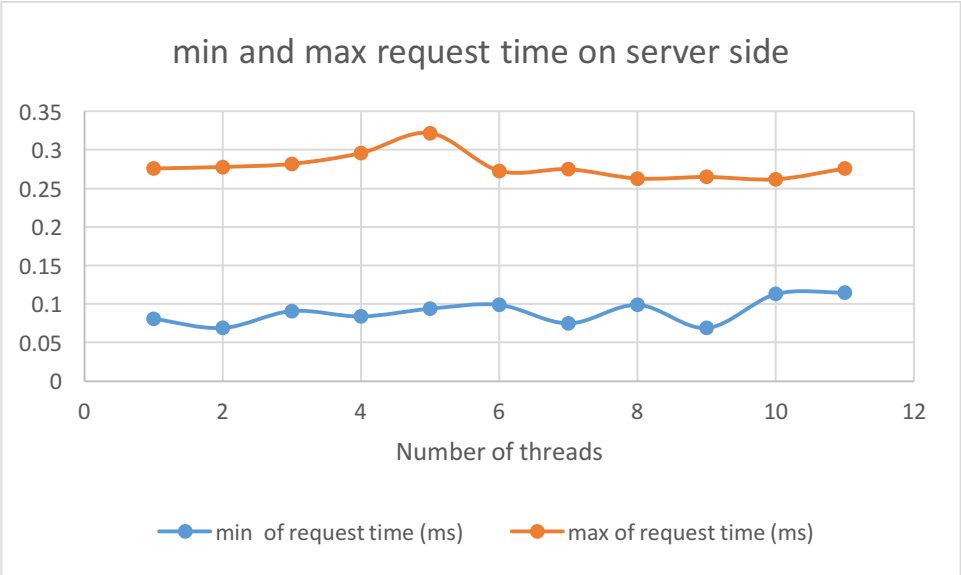
Things discovered:

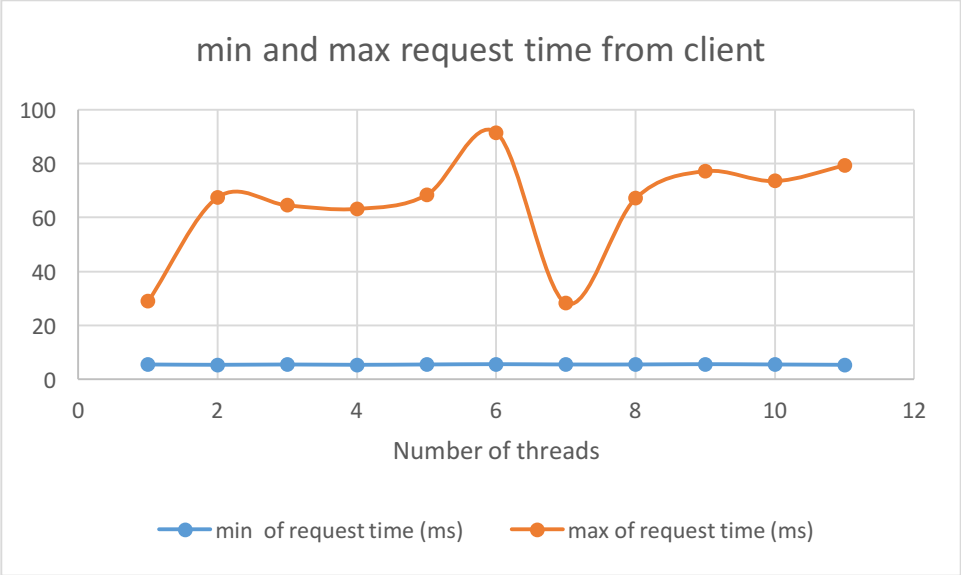
1. For server side, median and mean request time is largest when running on one thread. After thread number increased to 2, request time decreased dramatically. However, it doesn't decrease any more with more than 2 threads.
   Explanation:
   The server was run on a computer with four cores.
   I have two threads in the program keep running the server and thread pool, so the number of threads for input number of 2 is actually 4 threads. The request time on the server time reflect that 4 is the lowest best number of threads (including the other two threads) to run my program on a computer with 4 cores.





2. For the client request time, however, there isn't any relationship between the mean or median request time and number of threads. The possible reason is that the connection time is more than 10 times longer with respect to request time on server side, and we can only pass one session for each connection (sequential part takes too long time), so the number of threads won't affect request time from client.
3. From above charts, we see that mean/median request time from client is more than 10 times longer comparing to request time on server side.
4. From the charts below, we didn't see any relationship between min/max request time (from server or from client) and number of threads. However according to the median and mean request time from charts above, the overall performance of request time from server did get improved if the number of threads greater or equal to the number of cores (of the computer that runs the server).





5. Reusing connection is certainly a good choice to reduce the time wasted in handling new connection. The time spent on handling connection is much longer than request time on server side, so reusing the same connection should largely improve request time from client side.