Project title: I/O Completion based Server & Client Communication

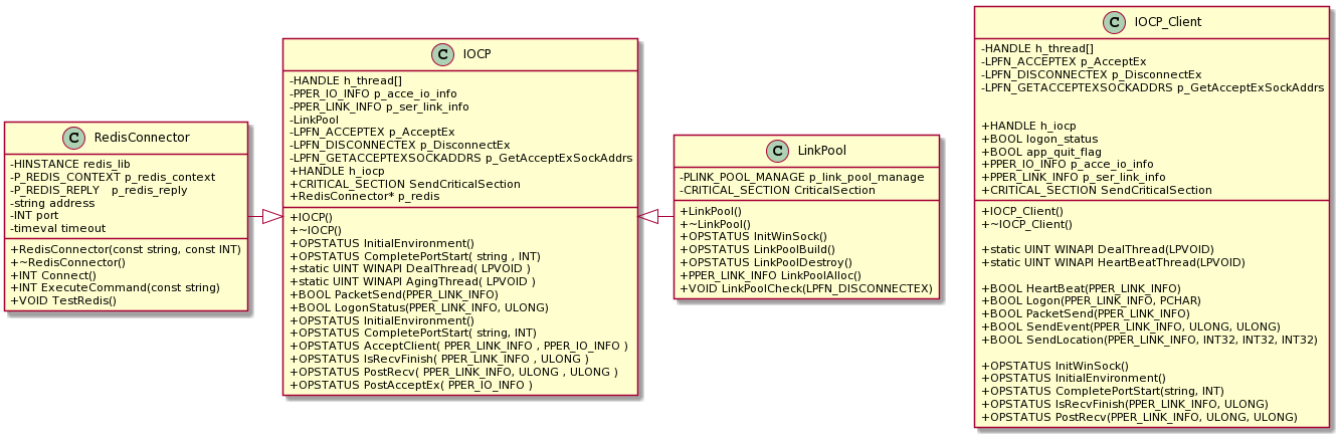
Team members: Leyen Qian (108384916), Ziyue Guo (107037369), Yi Hou (109288627)

Final State of System Statement:

We have done all parts of this project, it includes installing Redis server on a NAS server, compiling third part Redis connector C library, and a communication framework between clients and the server. Besides that, a simple test case has been implemented on the server side used to send commands to the Redis server. The only change compares with the project 5 is that we modified the interaction process between client and the Redis server, after the client send data to server, the server will form a command used to insert data into the Redis server. But the server will not redirect the Redis server's reply to the client side, because it is not necessary to show details to clients (IoTs). This change only can be seen from the sequence diagram, the class diagram is not affected.

Final Class Diagram and Comparison Statement:

UML class diagram

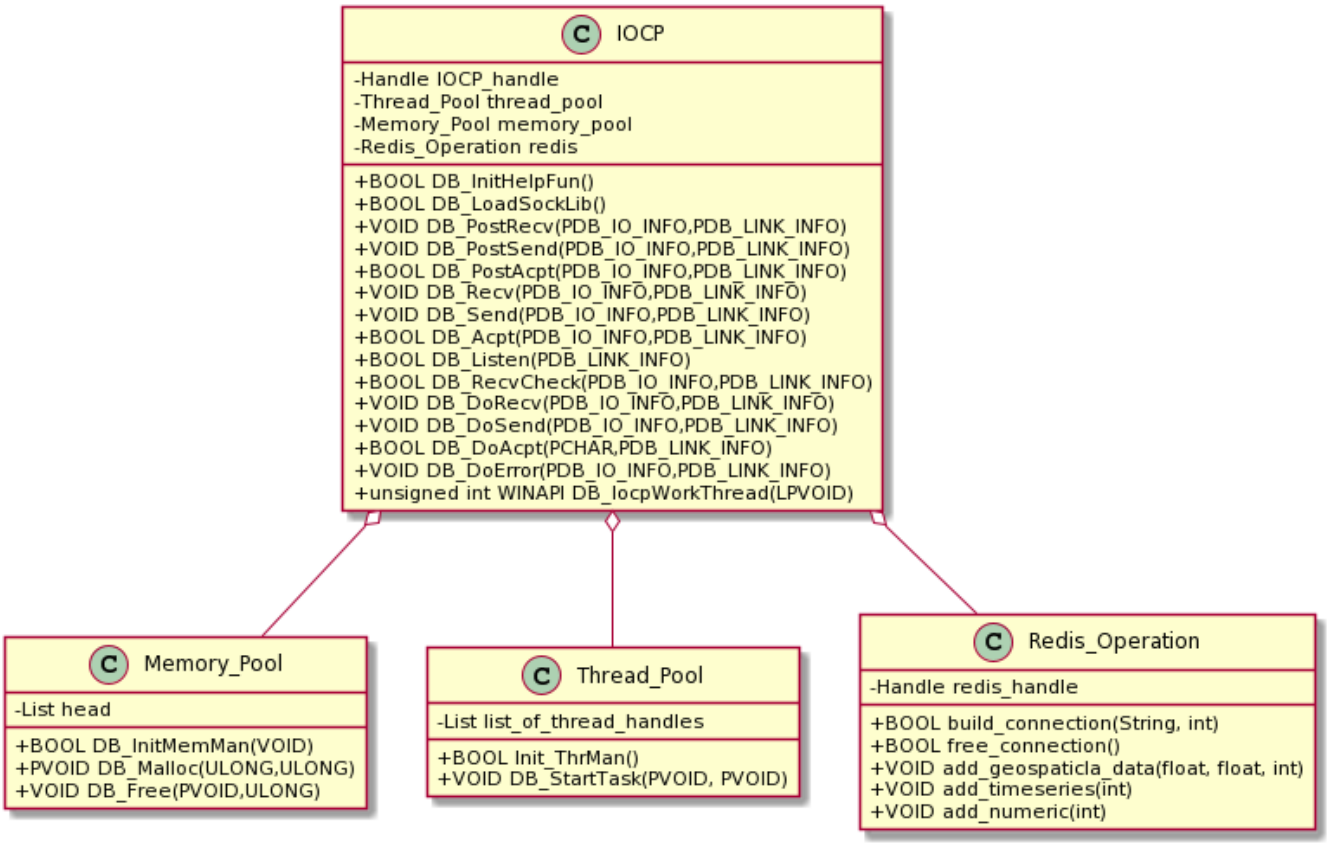


Command Pattern: between IOCP & IOCP_Client.

Object Pool Pattern: used in LinkPool

Façade Pattern: used in IOCP & RedisConnector (hide complex subsystem and provide simple access.)

UML class diagram from Project 4



The class diagram from the project 4 is different than that in project 6. In new diagram, we use link pool to take the place of memory pool. More specifically, the link pool includes all the functionalities of memory pool, besides that, it also performs basic initialization on the pre-allocated memory block, which reduces the latency of incoming client accept. Another noticeable modification is that we discard the design of thread pool and create fix number of message processing threads within the server instead. Another than that, only a few changes have been introduced in IOCP or IOCP_Client class, most are about message sending.

Third-Party code vs. Original code Statement:

Hiredis: https://github.com/redis/hiredis

Hiredis is a minimalistic C client library for the Redis database. We compile it into a dynamic load library (hiredis.dll). And use the library provided functions through several imported header files (hiredis.h for normal connection, hiredis_ssl.h for secure connection). More related third-party header files are located under the hiredis directory from the project. Besides, the method RedisConnector :: TestRedis is adopted from the example code of the hiredis library.

IOCP: https://docs.microsoft.com/en-us/windows/win32/fileio/i-o-completion-ports

IOCP (I/O Completion Ports) is an efficient threading model from Windows for processing multiple asynchronous I/O requests on a multiprocessor system. For the server and the client side, two system dynamic libraries, ws2_32.lib and Iphlpapi.lib, are being referred.

Statement on the OOAD process for your overall Semester Project:

Positive: modular design

test case for each module

plan and achieve consensus in design before coding

GitHub Repository:

https://github.com/LeyenQian/StreamProcessing

Demo Video (better resolution):

https://drive.google.com/file/d/1X9lN--BduCSryQfQPhAqcffMnsQY2TRq/view?usp=drivesdk