The Server Development Framework Based on the Completion Port

Junxi Yu
College of Information Engineering
Hebei United University
Tangshan, China
yjxmfc@163.com

Guohuan Lou
College of Information Engineering
Hebei United University
Tangshan, China
zdhua@heut.edu.cn

***Abstract*—The network service applications based on socket have been widely used, however it is still difficult to realize network service application program with a large amount of data and connections. This paper describes the principle of completion port, and based on this, analyses and introduces the various steps to achieve completion port and describes the usage of key functions in detail. Finally the client and server are simulated for the testing. The results show based on the I/O completion port technology the server application can get a good expansibility.**

Keywords- server ; client; complete port; thread poo

I. INTRODUCTION

With the rapid development of the Internet, the ever-changing network communications applications appear constantly in our work and life to meet the different needs of the various aspects. At the same time, these changes bring forward increasing demands to the server development, especially to robust, high-performance communications applications demand. The server is a software product that provides resource management and services for the client. The server must have the ability to be charged with services and guarantee services and this ability behaves as: high-speed computing power, stable running for long time and powerful data-handling capacity. With the more and more client amount, server end endure austere performance test and load pressure test[1]. Because server has the characteristics described above, the primary programmer are unable to do the work. Every time to engage a new server project development, it should be done to rewrite socket communication code as well as complex I/O management and task scheduling program repeatedly, and also the tedious test job with uncertain cycle should be faced. All these are not only a waste of time and money, but also can not guarantee the time limit for a project.

II. USER INTERFACE OF SERVER FRAMEWORK

The framework program exists in the form of dynamic link libraries. It can isolate third-party applications, third-party applications only need to complete the data logic-process part, and provide a unified programming interface for other IDE programming tool. In order to make user programming easier, server framework changes creatively the multi-task programming way into message event-driven programming way to the user. This feature is mainly due to the object-oriented programming ideas of C++. It is presented to user through the virtual function interface inside abstract class[1].

III. ANALYSIS OF KEY PROBLEMS FOR SERVER FRAMEWORK

A. *Thread pool technology*

In order to ensure the efficient operation of server, at the same time, to guarantee it will not occur that due to too many tasks are opened so that the server crashes, it is the best way to create a thread pool[2], that is, the limited threads are put together to wait activating data service for client, after service, returning to thread pool to wait a new data process task. The number of thread in the thread pool must be more than the number of CPU, so as to ensure that each CPU is assigned a thread (generally, there are several CPU in the server hardware configuration), and to maximize the efficiency of the entire server. Additional thread in the pool will stay there and does not consume the time of CPU. In case the new data arrive, the operating system will automatically activate a thread from the thread pool. This thread will be “woken up” to complete the receiving and sending data. After the thread has processed the data, it will automatically return to the pool and continue to wait the arrival of the new task. The thread pool technology not only can solve the spending on creating thread every time, but also enable each CPU as busy as possible, at the same time, it also does not consume too much time of CPU though thread context switch is needed. This scheme is the best server model at present. But there is also a problem to be solved for this server model, that is, the number of threads in thread pool must be reasonable, not the more the better. So the setting of thread number inside the thread pool should take into account the real application needs, it is generally related to the numbers of the CPU.

B. *Overlapped I / O data structure*

Completion port makes use of OVERLAPPED structure to realize overlapped I/O operation. If an overlapped I/O operation is completed, the OVERLAPPED structure must be provided with, that is, a pointer that points to the OVERLAPPED structure should be included in its parameters[3]. As we stated above, after operation end, this pointer can be carried back. But depending on only the OVERLAPPED structure pointed by this pointer, the application can not distinguish which type of operation has been completed. However, based on the above understanding, you only need to customize an extended OVERLAPPED structure and add the necessary tracking information in it. When the operation ends and a pointer pointing to an OVERLAPPED structure is obtained, CONTAINING_RECORD macro can be used to take out the base address pointing to the extended structure. Thereby the type of operation can be determined by reading the tracking information inside.

C. Identifing and search of client information

In completion Port applications, a large number of clients I/O requests must be dealed with. When a large number of clients are connected to the server and an asynchronous I/O operation successfully completed, in order to process the operation results the service thread must know that this I/O comes from which client and where the I/O data store (e.g. the client sockaddr_in structure data corresponding this Socket, the buffer storing network data etc.). Map serach method is given in reference [4], that is, to create a client underlying communication object which is corresponding with the socket one-to-one and the object is put into a container similar to map. According to the socket value, the container search algorithm is used to find its corresponding object information when needed. In the case of fewer client connections this method is not superior to the vector or deque method. However, when a large amount of clients are connected and the table is frequently looked-up, this method can improve efficiency to a large extent.

D. The key technique API for Completion Port (IOCP)

(1) HANDLE CreateIoCompletionPort(HANDLE FileHandle，

HANDLE ExistingCompletionPort，

ULONG_PTR CompletionKey，

DWORD NumberofConeurrentThreads)[5]；

Where, ‘FileHandle’ is associated with the completion port handle. ‘ExistingCompletionPort’ is an existed port handle, ‘CompletionKey’ is generally used to store the context information corresponding with ‘FileHandle’. ‘NumberofConeurrentThreads’ is the number of concurrent threads which are allowed to operate to this completion port. This function has two kinds of usages when a new completion port is created. ‘FileHandle’ can be NULL, the return value is the handle of the new port. And when an existed handle is needed to associate with port, ‘ExistingCompletionPort’ is port handle, ‘FileHandle’ and ‘CompletionKey’ are socket handle and the context information that corresponds with this socket. The working threads is twice of CPU in number that is contained in the operational environment in which applications run.

(2) BOOL GetQueuedComPletionstatus(HANDLE ComPletionPort，

LPDWORD lpNumberofBytes，

PULONG_PTR lpComPletionKey，

LPOVERLAPPED *lpOverlapped，

DWORD dwMilliseeonds)[6]；

By calling the ‘GetQueuedCompletionstatus’ function, the socket information can be achieved when events occur. The transmitted byte amounts can be obtained with ‘lpNumberofBytes’ parameter. The context information related with the socket handle can be obtained through the ‘lpCompletionKey’. This context information is the third parameter ‘CompletionKey’ data in the function ‘CreateIoCompletion’. The overlap I/O parameter address used in the I/O request can be obtained by the ‘lpOverlapped’ parameter.

(3) BOOL PostQueuedCompletionStatus(HANDLE CompletionPort,

DWORD dwNumberOfBytesTransferred,

ULONG_PTR dwCompletionKey,

LPOVERLAPPED lpOverlapped)；

The threads in the thread pool can be notified to exit by calling ‘PostQueuedCompletionStatus’ function. ‘CompletionPort’ is the IOCP kernel handle, ‘dwNumberOfBytesTransferred’ parameter is the amount of bytes to be transferred. ‘dwCompletionKey’ is the context information corresponding to ‘FileHandle’, ‘lpOverlapped’ is overlap I / O sent to the ‘FileHandle’

E. Management technology of memory pool

This idea is from the memory management of operating system. If the programs apply for and release memory frequently, operating system will leave a large number of memory fragmentation in its internal heap. If so running for a long time, the speed that programs apply for and release memory at will be more and more slow and excessive memory fragmentation also has a serious impact to the stability of the server program. When the server processes the huge amount connections or asynchronous I/O operation, enormous memory sources are required. The capacity of the memory pool is a problem that all server program developers pay attention to. The various factors should be considered for the size of capacity. If the size of memory pool is too large, the operating system resources will be wasted. If the size of memory pool is too small, applying the new memory resources to the operating system will occur frequently. So its value must be appropriate and be decided based on the actual server operation.

IV. COMPOSITION OF THE SERVER FRAMEWORK

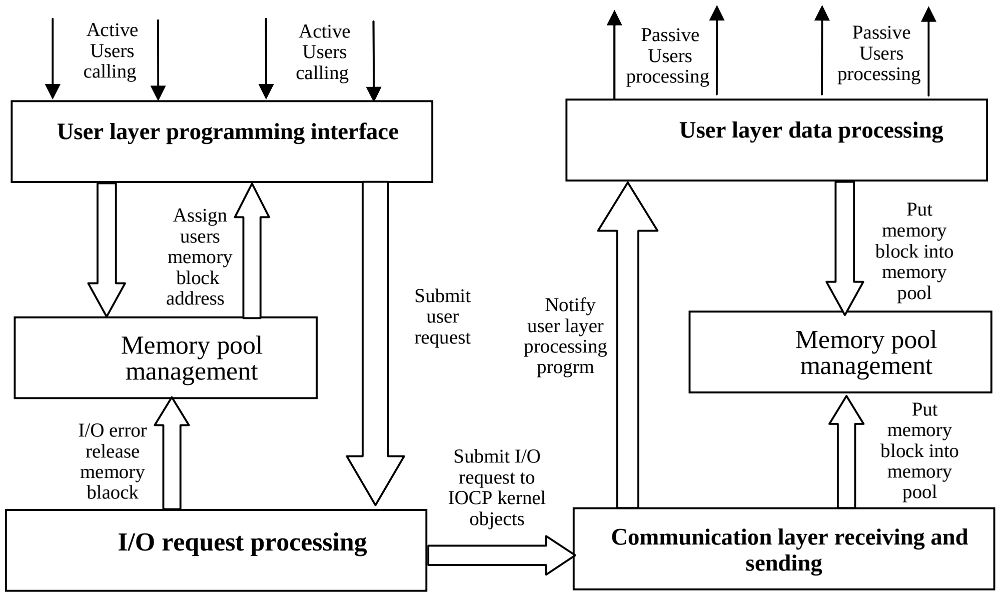
Active Users calling
Active Users calling
User layer programming interface
Assign users memory block address
Memory pool management
I/O error release memory blaock
Submit user request
I/O request processing
Submit I/O request to IOCP kernel objects
Passive Users processing
Passive Users processing
User layer data processing
Notify user layer processing progrm
Put memory block into memory pool
Memory pool management
Put memory block into memory pool
Communication layer receiving and sending

Figure 1. The composition of server framework

As shown in Figure 1, the active user calling represents that the third-party program thread calls a standard programming interface provided by this server framework, drives internal function unit of the server program to work. While passive user processing means that this framework structure will call a third-party program (This progress program is coded by user) . If user has a request to send data, user layer programming interface will be called, the request of user is submitted to the I/O request processing layer. Then I/O system calling is issued by the I/O request processing layer. Communication layer receiving and sending is responsible for handling I/O requests work and submit this request to the user layer processing, finally, this layer calls the user processing program to complete the data processing.

V. TEST ENVIRONMENT AND RESULTS

Test environment is as follows: A computer with a dual-core Genuine Intel CPU T2080 1.73GHz, 1024M memory, a computer with a dual-core Intel Core i3-380M 2.53GHz, 2048M memory and a computer with a dual-core Intel Cool Core i3-2330M 2.2GHz 2048M memory. A PC machine has an operating system of Windows Server 2003 and other machines have the operating system of Windows XP. The virtual client is installed in one of the PCs, it is used to simulate a large number of clients. One computer is used to simulate the real client and another PC is used to simulate server.

When testing, 100-2000 virtual clients are started respectively, which represent that a different number of clients access the server at the same time. Figure 2 illustrates the experiment results.

During testing the server is at high load environment. First client connects server successfully, the client sends data to ask server providing services for it. After receiving data the server analyses data and then sends the response to client. Next the client sends a request to server again, and all of the client and server use this kind of communication and this process repeats continually. Figure 2 shows when the number of client changes from 100 to 2000 request-response delay varies linearly with the growing client, which shows that it is stable for server to handle I/O requests. Figure 3 shows that the memory consumption of the server increases with the growing number of clients.

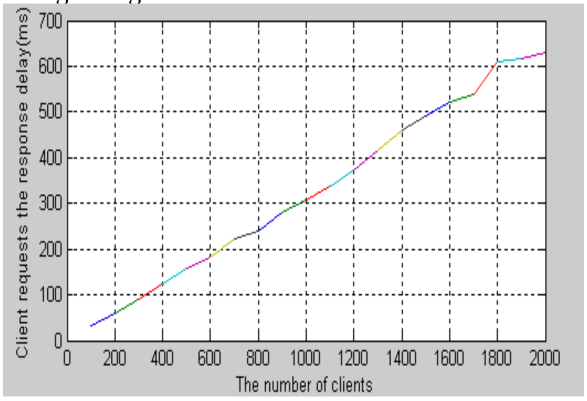

Figure 2. The response delay for client requests

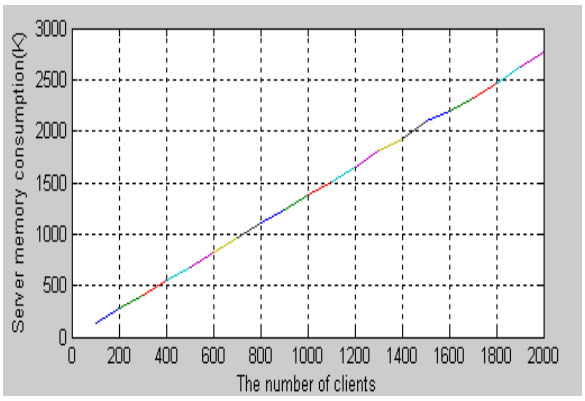

Figure 3. Server memory consumption

VI. CONCLUSION

Based I/O completion port technology, the server application can get a good expansibility. This paper describes the work principle of completion port, and based on this, analyses and introduces the various steps to achieve completion port and describes the usage of key functions in detail. IOCP programming interface is provided in Microsoft Windows Server 2003 and Windows Server 2008 and this technology is used to handle customer requests in IIS, the better results are obtained. It can be predicted that there will be more and more server application technology.

REFERENCES

[1] Wu Xing， Huang Ai ping ，“Implement the Scalable Server Application with Completion Ports,” Computer Science， 2002， 29(11)：pp.144-146.

[2] Zhang yuan， Cui bin,Yu Xiaoqing . “Application of Thread Pool Technology in the network game server.” Control & Automation， 2006， 22(6), pp.42-44.

[3] Liao hong jian， Yang yun bao， Tang lian zhang. “Key issues of high-performance server communication layer by using I/O completion port.” Computer Applications， 2012， 32(3), pp.812-813.

[4] HOLGATE L.“Handling multiple pending socket read and write operations.” [EB/OL]. http://www.codeproject.com/KB/IP/ reusablesocketserver4.aspx.

[5] Fan Song, Lu Ying, Ma Liquan. “Method and its implementation for I/O completion port network server performance improvement.” Journal of Dalian Institute of Light Industry. 2006,25(4), pp.282-285.

[6] Microsoft. I/O completion ports [EB /OL]. http://msdn.microsoft.com/en-us/library/windows/desktop/aa365198 (v=vs.85).