A Software Architecture Based on High-performance Middleware

Jiameng Xie [1,2], Hong Peng [1], Donghui Yang [2], Zhonghua Zhu [3]

[1] College of Computer Science and Engineering South China University of Technology, Guangdong Guangzhou, 510641, China

[2] Traffic Administration Bureau of Guangdong Province, Guangdong Guangzhou, 510440, China

[3] Jingma Science and Technology Corporation of Shenzhen, Guangdong Shenzhen, 520000, China

Abstract

Software architecture is the base of software design, and it has various implementations in different applications. In China, with the development of computer and network technology, more and more governments' applications based on network are set up, which have many differences to ordinary enterprise applications. In this paper, through analyzing the characteristics and demands of governments, a demultiplexing connection and reactor/thread priority architecture based on high-performance CORBA middleware is set up. A realization in traffic administration bureau of Guangdong province is provided as an example. It gives a feasible solution to software design of specific government applications.

Keywords: Software Architecture, Middleware, Model

1. Introduction

In recent years, with the development of high-speed computer networks, more and more applications based on network are built. Accordingly, software architecture is designed for distributed systems other than centralized systems, which used in previous days. At the meanwhile, various techniques, such as object orient technique, middleware technique, distributed computing technique, etc, are widely used, which give strong support to software design. In China, the design of government's information system has attracted more and more attentions. Nowadays, many macro information systems are set up for different government's administration. There are many similarities among these applications. Therefore, in this paper, we choose traffic administration information system, which designed for the traffic administrator bureau of Guangdong province of China, as an example. Through analyzing its characteristics from the view of software engineering, we set up a demultiplexing connection and reactor/thread priority architecture based on high-performance CORBA middleware. It gives a feasible solution to software design for specific government applications.

This paper is organized as follows. Section 2 discusses characters of information system of government in China. Section 3 presents a software architecture based on high-performance CORBA middleware. Section 4 gives an application used by traffic administration department as an example. Section 5 provides a conclusion.

2. Characters of Traffic Administration Application

Traffic administration bureau is one of the most important branches of Chinese national public security department. It takes charge of the managements of vehicles, drivers, traffic violations, traffic accidents, and so on. Its information system is one of the most principal applications of Chinese government. It is a real-time distributed transaction processing system, which based on WAN (Wide Area Network). Its main functions are to execute various kinds of traffic administrations. Since its characters are typical in government applications, we discuss them as below:

Firstly, the system can be logically divided into respectively hierarchical "data centers". Owing to the administration system of China, which is the system whereby authority passes down from the top through a series of executive positions in which each is accountable to one directly superior. Therefore, its structure is like a tree, i.e. every city's administration department sets up basic transaction database as the leaf node, and their father nodes are the data centers built by province's governments, which data are sum up from the databases of cities. The root of the tree is the national data center. Different cities (leaves) exchange information through the corresponding data centers (father nodes), which set up in different hierarchy. All data centers are connected together to form an integrated transaction flow. The processing ability of data center must be strong enough to conduct real-time processes and reliable communications.

Secondly, since the system is based on WAN and used across a vast territory, it demands to support

various physical network connection modes, including traditional modes and wireless techniques, such as GPRS, CDMA, etc. In addition, the system is required to be sensitive to environment; and problems about time lag, network bandwidth, and security must be solved effectively.

Thirdly, the system comprises diverse types of applications; moreover, within the lifecycle of software, the content of application will always be adjusted according to the traffic policies. So, the system is required to be high flexibility and facile maintainability. Furthermore, an effective routine maintenance method must be set up to reduce cost of maintenance.

Fourthly, because system uses "client-server" model, so the object-orient middleware technique, which based on three-layers architecture is required. Moreover, specific software architecture for such special field is demanded.

All in all, in order to improve system's expansibility and maintainability, object-orient technique is used. In order to improve performance and reduce usage of system's resource, three-layers "client-server" architecture based on high performance middleware is used. Therefore, how to choose middleware products and design its architecture is one of the most important problems. In this paper, a design project based on high performance real-time middleware CORBA is provided, and the connection and concurrent mechanisms are expatiated.

3. Software Architecture based on High Performance Middleware

Nowadays, CORBA, Sun Java/RMI (J2EE), Microsoft DCOM/COM+ are the most popular distributed object models used in large-scale applications. In fact, different techniques are suitable for different kinds of applications, i.e. DCOM is fit for work groups applications; JAVA/RMI adapts to internet applications; CORBA which support multi-operation system, various kinds of network and business suits enterprise applications, and there is less technical risk in software design for large enterprises applications. So, CORBA and improving on the connection and architecture of thread are chosen to make an integrate client/server scheme.

3.1. Architecture of CORBA Logical Server

Figure 1 shows the general organization of a CORBA system[2].

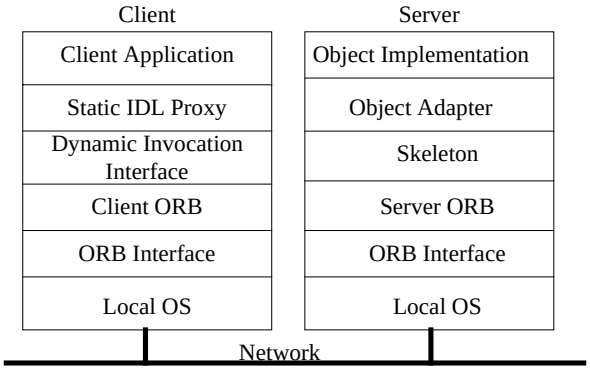

Fig. 1: General Organization of a CORBA System

CORBA is organized as a collection of clients and object servers. The ORB is responsible for handling the basic communication between a client and an object. The object adapter takes care of forwarding incoming requests to the proper object. The skeletons takes care of unmarshaling.

Figure 2 shows the architecture of CORBA logical server.

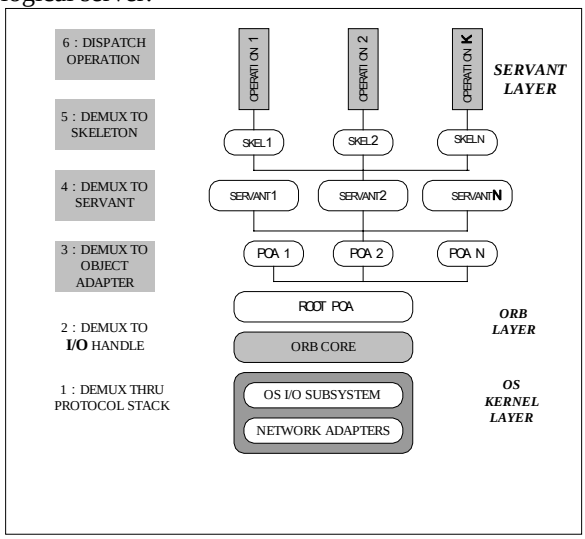

Fig. 2: CORBA Logical Server Architecture

The operations are list as follows.

Step 1 and step 2: OS protocol stack dispatches client's request from low layer to high layer. The request passes through network interface layer, data-link layer, network layer and transport layer in sequence, at last, it reaches user/kernel boundary, i.e. socket layer.

Step 3 and step 4: the kernel of ORB uses address information of client's object key to locate appropriate POA (Portable Object Adapter) and servant.

Step 5 and step 6: POA uses operation name to find out appropriate framework of IDL (Interface Define Language). IDL framework divides the buffering contents into operational parameters, and then does the operations.

Such dispatch mechanism of ORB, which strictly according to functions cannot fit demands of high

performance application. The reasons are list as follows.

Firstly, efficiency is low. The main reason is that it costs high to dispatch requests of client hierarchically, especially, when IDL interface has many operations or an object adapter manage more than one servo.

Secondly, it causes reverse of priority and uncertainty of system's response time. Using such hierarchical architecture, client's requests are processed strictly according to FIFO order. Consequently, request with high priority may be managed after which with low priority. If response time of low priority request is uncertainty, it causes uncertainty of response time of request, which priority is high but its management priority is not high. Therefore, it can't satisfy demands of high performance applications, which require strict response time.

The main function of ORB core is to execute GIOP protocol through establishing connection and realize concurrent. So, the architecture of connection and concurrent is determining element of ORB core. In order to improve performance of ORB core, suitable software architecture must be set up to manage connection and concurrent.

3.2. Architecture of Multiplexing Connection

Multiplexing connection is the most common connection architecture. It can realize extensible applications through reducing the numbers of TCP connections. However, the most challenge of such architecture is to solve synchronized operations of reading and writing.

3.2.1. Active Connection Architecture

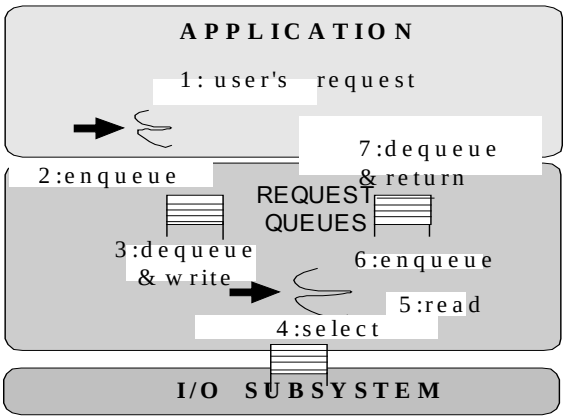

Fig. 3: Active Connection Architecture

As shown in Figure 3, such architecture bases on multi-thread model of producer/consumer. The operation steps are: (1) Application-thread pushes "operation" into the queue. (2) Write-thread takes "operation" away from the queue, and then sends "write" instruction to the socket. (3) Read-thread reads from socket, and then pushes "return" into the queue. (4) The application takes "return" away from the queue.

This architecture is the most typical multi-thread architecture. The main advantage of it is facile realization. The shortcoming is low efficiency causing by high cost of "dispatch" operation.

3.2.2 Architecture of Leader/Follower Connection

The operation steps are listed below: (1) the application-thread executes "write" operation directly. (2) The thread, which executes "write" operation and wait for response, executes "select" operation, and then waits for response on the socket. This application-thread is called "leader". (3) The thread following "leader" and waiting for response will jam on the token managed by ORB core. (4) Once the response returns to client, the "leader" reads this response, and looks for owner by using serial number of GIOP head. If owner is "leader" itself, it releases token to "follower" and returns. Then follower becomes leader. Otherwise, if the response does not belong to leader, it informs its owner to get-response, and continues to jam on the socket.

The most significant advantage of this architecture is that it improves efficiency through decreasing quantities of "dispatch" operation. The shortcoming is the complex logic of realization and lock mechanism, which causes great additional costs.

3.3. Architecture of Demultiplexing Connection

This architecture is designed for the application discussed above. See figure 4.

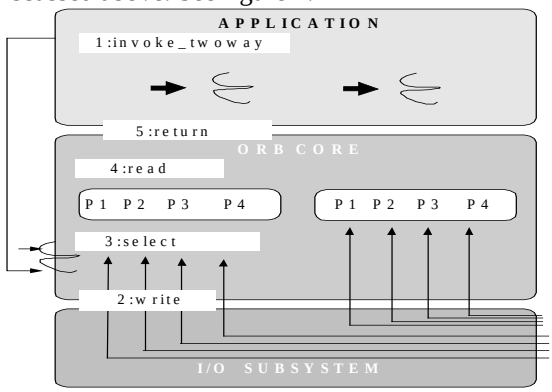

Fig. 4. Demultiplexing Connection Architecture

In this architecture, the TSS (Thread Specific Storage) of client-thread holds linker-table of server created beforehand. In one priority, every thread holds its own link respectively to ensure that different threads needn't have to share same socket. All operations, such as write, read, select and return, haven't any problems of competition.

The advantage of such architecture is that it can keep end-to-end priorities. It hasn't any competition of resource and has low cost of synchronization. The shortcoming is that it increases amounts of connection. So, it fits for application of static configuration i.e. application with few priorities and an invariable quantity of priority. Considering applications of traffic administration, since the most important characters of it are that the variety of operations is limited, and priorities of each operation are few. Therefore, it is most suitable for traffic applications.

3.4. Architecture of Thread Pool

In order to cooperate with demultiplexing connection architecture discussed above, a reactor /thread priority architecture is designed. See figure 5.

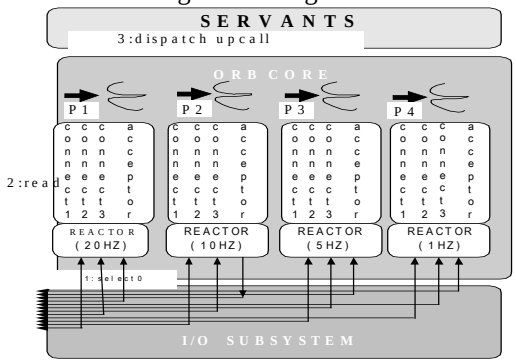

Fig. 5. Reactor/Thread Priority Architecture

This architecture makes threads, which have different operational priorities connect with a group of reactors. In every thread, the reactor operates: (1) dispatches requests coming from clients, (2) reads these requests and dispatches them to corresponding daemons to finish operations.

The advantages are that it can reduce not only the shifting quantities of context, but also the costs of lock processing. It can lock statements of daemon within different threads, which have different priorities. It supports scheduling strategy and analytical technique, which make request's processing rate connect with its priority. The disadvantage is that in every thread, it deals with client's request in sequences, thus depresses the parallelism of system.

4. Application Example

As an example, we set up an information system for the traffic administration bureau of Guangdong province of China. In this system, demultiplexing connection and reactor/thread priority architecture, which based on high performance middleware CORBA, are used to build an integrate client/server system. UNIX/Windows NT is chosen for database server, and Windows NT/2000 is used as application server. OS of clients are Windows 98/2000/XP. The database is Oracle 8/9i. The middleware is CORBA. Software architecture of this application system is shown in figure 6.

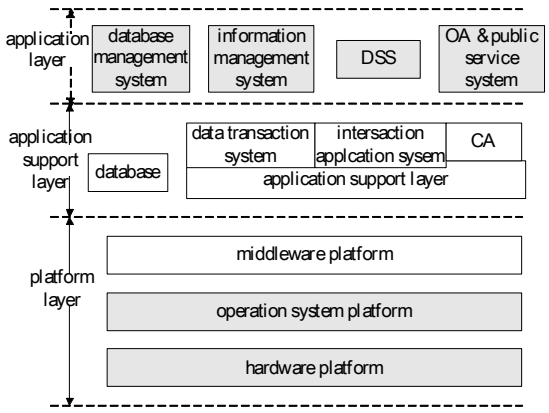

Fig. 6. Software Architecture of Traffic Administration System

5. Conclusions

In this paper, characters and demands of traffic administration application in China are analyzed, and then a demultiplexing connection and reactor/thread priority architecture, which based on high performance middleware CORBA, are designed. As an example, such architecture is used to set up traffic administration information system for government of Guangdong province. The system has a good performance in practice. Further research works will include modifying shortcomings of this architecture, improving system's agility and ability of parallelism.

6. References

[1] Mary Shaw, David Garlan, "Software architecture: perspectives on an emerging discipline," *Prentice Hall Inc*, 1996.

[2] Andrew S. Tanenbaum, Maarten van Steen, "Distributed Systems—Principles and Paradigms", *Prentice Hall Inc*, 2002.