Study on CORBA-Based Load Balance Algorithm*

Gongxuan Zhang[1], Jianfang Ge[2], and Changan Jiang[1]

[1] Department of Computer Science and Technology,
Nanjing University of Science and Technology,
210094 Nanjing, China
Gongxuan@mail.njust.edu.cn
[2] Department of applied mathematics,
Nantong Institute of Technology,
226007 Nantong, China

Abstract. With the increment of the number of Internet applications, in order to solve some problems such as heterogeneous access, dynamic services and their migration, a number of middleware and load balancing methods have been developed. CORBA is a standard of application middleware made by OMG. After studying some load balancing algorithms, we present an improved, CORBA-based load balancing model and its migration algorithm.

1 Introduction

In recent years, the Internet has been applied over most of our real world. Companies use it for their businesses to provide customers with their products or services. With the explosion of the number and type of services, some new mechanisms and frameworks are required to meet the needs of customers with ease. The mechanisms and frameworks are developed to discover, invoke and migrate web resources. Many new technologies are developed to improve business processes and quality of services over Internet.

CORBA, Common Object Request Broker Architecture, is a standard of middleware gave by Object Management Group[2]. There are many specifications of object services, for instance, naming service and trading service, described in the standard. With naming service, a client computer can locate an object reference by symbol name. And with trading services, a client is able to take advantage of the service's dynamic discovery mechanism for the object location. The mechanism is called dynamic binding. Trading services are hosted on a computer, with providing query service descriptions that clients can dynamically invoke the services just as the yellow pages of telephones. The proposed CORBA based load balancing model (in brief, CBLB model) is based on CORBA trading services. The rests of this paper are organized as follows: Section 2 describes trading service and layered load balancing abstract. Section 3 discusses the CBLB model, Section 4 shows migration algorithm. And the last is the conclusion.

* This is partly sponsored by the National Science Fund Committee, and the project number is 40120140817.

© Springer-Verlag Berlin Heidelberg 2003

2 Trading Service

A CORBA Trading Service provides the function of detecting a dynamic object, and the client can ask it to locate the object. Similar to the naming service, the trading service stores object references. Note that, the trading service does not save the reference's name but saves the service description that the reference provides. The follow are some concepts about the object trading service:

1. A bulletin, used for trading storage, is called a service offer that contains the service's description and an object reference that provides the service. The service offer is still of some concrete service types.
2. The action of registering a bulletin is called 'export', and the actor of exporting is called to an exporter. That is a server's action.
3. The object reference inside the service offer is an object of which provides bulletin service, that is a service provider which can't be changed after the service offer is imported. And before the object reference does not remove and the service offer is imported again, the object reference can't be changed either
4. Inside a service offer, its service description, of which means the bulletin's "original text", is constructed by many name-value pairs. Compared to service supporters, the values may update. The same service promoters may be repeatedly advertised, but the values are not the same. Many bulletins have same name-values but different service promoters. This compare to an advertisement can list many stores in different areas
5. Bulletins can be withdrawn from the trading service. The bulletins are withdrawn or deleted only by servers.
6. The action for standard services to search for bulletins is called 'import', and the actor is an importer. That is a client's action.

In practice, we can take the Trading Service as the trading platform between CORBA object developers and CORBA clients. The developers register, explain and promote their objects through the application interfaces or with tools provided by object trading services, and the clients query, search, get and invoke the objects.

3 CBLB Model

In the CORBA environment, there are several layered abstracts for load balancing classification in order to construct a practical system. They are object level load balancing policy that consists of several CORBA objects identified by object reference, object implementing level load balancing policy that load balancing is implemented through active entities, for instance, processes of operating system, and system level load balancing policy that

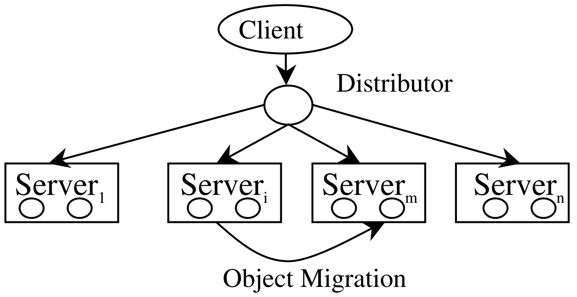

Fig. 1. Basic load balancing model

processes are distributed to different computing nodes just like traditional load balancing.

In this paper, the first two policies are introduced. With object level load balancing, an object reference will be taken if another object send a request to it. The client's requests are distributed to available servers. A client can contact the trading server and obtain appropriate object references if he needs specific services. The load balancing actions are also done within a trading server, and the server selects suitable object references and then sends the references back to the client. Finally, the client choices the object references and the object services.

With object implementing level load balancing, objects are special inner object instances of processes. The load balancing is implemented by distributing the instances' services uniformly to servers' processes, or by migrating the instances' services. As depicted in figure 1 is the basic way for load balancing.

As all known, the load balancing strategies are more complicated in a client/server environment. The client carries out a series of operations, each operation may send a request to a specific service. And many servers provide services together to meet all the requests. So the client must find out the servers that can handle its request.
The main responsibility of a trading server is to delivers services between the client and the server. By a trading server, a client can dynamically and accurately find out the servers that provide the services. The basic steps are shown in figure 2. As service-exporters, the servers export the services. This means they register their services in a trading server. As service-importers, the clients first invoke some operations of a trading server and acquire the information of a server's services. And then the clients can directly send its requests to the servers and invoke the services. The server can remove its registered services from the trading server by invoking the function withdraw() to the trading server.

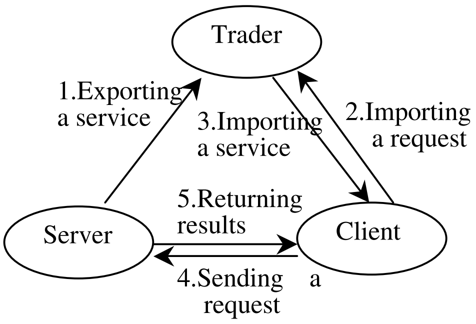

Fig. 2. Steps of services delivery

The important problems are that how services are registered up to the trading server (called trader below) and specified by the clients. The solution of CBLB model is that a server must give its service description by specifying some special properties or service functions, and exports the description for clients to acquire the service. This means that a client must specify the name of a service type when importing a service, and then the trader maps the type to a corresponding object reference and sends the reference back to the client. It is very convenient for the client to communicate with the related servers. In the CORBA environment, the trader can be transparently embedded into an ORB (Object Request Broker) as one of basic CORBA services. In this paper, the trader is also a component used for the clients and servers.

In our proposed model, the mission of a server is to determine what resources are available for a client. A server is a process running in a specific host with the CORBA environment. For the sake of the convenience of description, we suppose each host circulate a server only. The available server can handle with a series of requests from the client. And the trader is to deliver the services between clients and servers. In addition, a supervisor is designed to deal with the service objects' migration among servers. So there are two aspects to deal with the client's requests to some servers with balance: one is by the trader's allocation policy and the other is by the monitor's migration algorithm.

There are three kinds of nodes in the CBLB model. The server node exports services to the trader node for registration. The monitor or the supervisor, as a special server, gathers load balance information from the server and then migrates some services to different servers if necessary. On the other hand, the client node imports available services from the trader and then invokes the services. When receiving a client's 'importing' request, the trader must quickly search the specific services by executing the function: t:T→ρ(S). If the service type Ti is provided by the current servers {Si1,Si2,....Sik}, the function's result is t(Ti)= { Si1,Si2,....Sik }. Of course, the function t is dynamically affected when the server withdraws the registered services or the monitor migrates services among the servers.

4 Migration Algorithm

As stated above, the service migration is the monitor's job. So the monitor must gather load balance information from the servers and then make a decision whether the service objects are migrated. Once making the decision, the monitor starts a migrating process. During the migration, he announces the destination server to immigrate the specific service objects from the source server. A basic implementation for migration is that the destination server creates a new service object as the same type one as in the source server and then receives the objects' status from the source server. Forwards, the destination server announces the source server receives the new object reference in reverse. At the end, the destination server announces the trader that the object has been migrated correctly and asks the source server to remove the object. The entire procedure is depicted in Figure 3 and the following two cases should be considered if a client sends a request to a server while the server is doing migration:

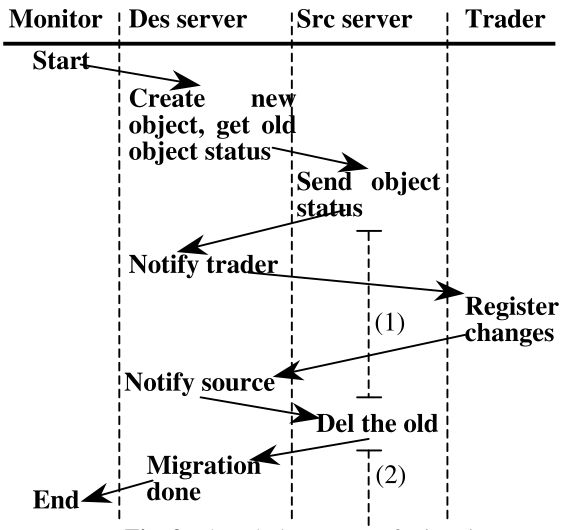

Fig. 3. The whole process of migration

(1) One case is that the source object is not removed just after its states have been migrated when the request comes to the source server. The solution is the source server sends the new reference to the client and then the client sends the request to the destination server.

(2) The other is that the migration has finished and the old object has been removed when the client sends the same request. The client will get error information and must import service objects again from the trader. In this case, the client invokes the new object reference from the trader.

Just as the discussion, a client always acquires correct service object references.

For the load balance algorithm, Peter Scheuermann's idea [4], with which the file is moved from a 'hottest disk' to a 'coolest disk' (called 'disk cooling process'), is introduced to our service object migration that the server stands for the disk and the service object for the file. The task of the migration algorithm is to move the service object from a 'heavier load server' to a 'lighter load server'. This is called 'the cooling server migration algorithm'.

5 Conclusion

With the CBLB model and the migration algorithm, some tests have been taken in a communication network with 7 computers. The performance is better but the trader will become a neck-bottle and block the migration when the service objects migration occur much frequently.

But there are two benefits in this proposed approach. First the CBLB model can easily join to existing applications as an ORB. The trader is embedded and taken as the ORB. Second the migration algorithm may easily be implemented in programming C++ or Java.

References

1. Andrew S. Tanenbaum. Distributed Systems: Principles and Paradigms. Prentice-Hall, Inc (2002).
2. K. Wallnau. Common Object Request Broker Architecture. Software Technology Review, Software Engineering Institute, Carnegie Mellon University (2000)
3. B. Schiemann, L. Borrmann. A new approach for load balancing in high-performance decision support systems. Future Generations Computer Systems, No. 12, Elsevier / North-Holland (1997) 345–355
4. Peter Scheuermann Gerhard Weikum and Peter Zabback. Disk Cooling Parallel Disk Systems. Vol. 17. No.3. IEEE Data Engineering Bulletin (1994) 29–40