Volume 2, Issue 7, July 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Design Challenges of Load Balancing-CORBA Architecture

Gagandeep singh
Deptt. Of Computer science and IT
GGS Khalsa College Sarhali, Tarn Taran
India

***Abstract:** In this paper we are going to discuss about the CORBA technique used in distributed systems and its various design challenges faced when adding this load balancing service to our CORBA ORB (object request broker) as network centric computing becomes more pervasive and applications become more distributed, the demand for greater scalability and dependability is increasing. Distributed system scalability can degrade significantly.*

***Key words:** Load balancing, replicas, portable, distributed*

I. Introduction

CORBA which stands for common object request broker architecture, is an industrial standard developed by OMG to aid in distributed programming. CORBA is just a specification for creating and using distributed objects. CORBA is not a programming language.
The CORBA [1] architecture is based on the object model. This model is derived from the abstract core object model defined by the OMG in the object management architecture guide. The model is abstract in the sense that it is not directly realized by any particular technology, this allows applications to be built in a standard manner using basic building blocks such as objects. Therefore, a CORBA based system is a collection of objects that isolates the requests of services (clients) from the providers of services (servers) by a well defined encapsulating interface. It is important to note that CORBA objects differ from typical programming objects in three ways:

1. CORBA objects can run on any platform [14].
2. CORBA objects can be located anywhere on the network.
3. CORBA objects can be written in any language that has IDL mapping.

II. CORBA architecture

CORBA is composed of five major components ORB, IDL, Dynamic invocation interface (DII), interface repositories(IR) and object adapters (OA)[5]

1. The Object Request Broker

The CORBA specification must have software to implement it. The software that implements the CORBA specification is called ORB. The ORB, which is the heart of CORBA, is responsible for all the mechanisms required to perform these tasks [3].

(a) Find the object implementation for the request
(b) Prepare the object implementation to receive the request
(c) Communicate the data making up the request

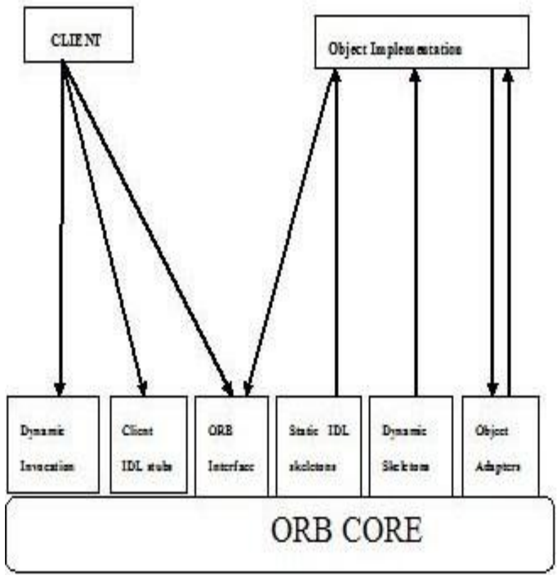

Fig1: - The structure of CORBA 2.0 ORB

There are two important things to note about the CORBA architecture and its computing model

I. Both the client[12] and the object implementation are isolated from the ORB by an IDL interface.
II. All requests are managed by the ORB. This means that every invocation of a CORBA object is passed to an ORB.

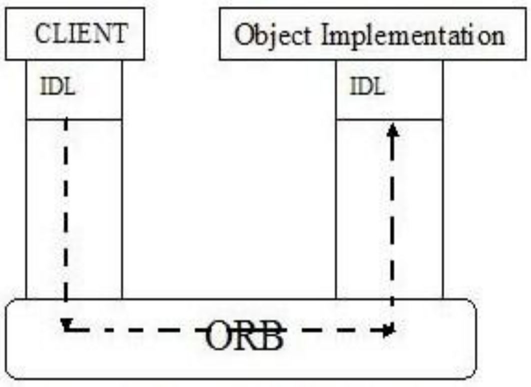

Fig2: - A request from a client to an object implementation

2. Interface Definition Language

The IDL defines the typed of objects by defining their interfaces. An interface consists of a set of named operations and the parameters to those operations. IDL is only used to describe interfaces, not implementations. Though IDL, a particular object implementation tells its potential clients what operations are available and how they should be invoked. Some of the programming languages with IDL mapping include C, C++, JAVA, LISP and PYTHON. Thus once you define an interface to objects in IDL, you are free to implement the object using any suitable programming language that has IDL mapping. [6][7]

3. Dynamic Invocation Interface

Invoking operations can be done through either static or dynamic interfaces. Static invocations are determined at compile time, and they are presented to the client using stubs. The DII, on the other hand allows client application to use server objects without knowing the type of those objects at compile time. [8] It allows a client to obtain an instance of a CORBA object and make invocation on that object by dynamically constructing request. CORBA supports both the dynamic and static invocation interfaces.

4. Dynamic Skeleton Interface

Analogous to the DII is the server side dynamic skeleton interface (DSI), which allows servers to be written without having skeleton, or compile time knowledge for the objects being implementation.

DSI was introduced in CORBA 2.0. its main purpose is to support the implementation of gateways between ORB's which utilize different communication protocols[20].

5. Interface Repository

The IR provides another way to specify the interface to objects. Interface can be added to the interface repository service. Using the IR a client should be able to locate an object that is unknown at compile time, find information about interface , then build a request to be forwarded through the ORB.

Here are the various design challenges which are faced the development of CORBA ORB load balancing service.

1. Implementing portable load balancing.
2. Enhancing feedback and control.
3. Supporting modular load balancing strategies.
4. Identifying objects uniquely.
5. Integrating all the load balancing components effectively.

Challenge 1:- *Implementing portable load balancing*

Changing application code particularly client applications to support load balancing can be tedious, error prone and costly. Changing the middleware infrastructure to support load balancing is also problematic since the same middleware[11] may be used in applications that do not require load balancing. Using ad-hoc or proprietary interfaces to add load balancing to existing middleware can increase maintenance effort.

Challenge 2:- *Enhancing feedback and control*

Sampling loads from replicas should be as transparent as possible to the replicas. If load sampling was not transparent, a load balancer would have to sample loads from server replicas directly, which is undesirable since it would require replicas to collect loads. If replicas collect loads, however application code to support load balancing. A load balancer should not be tightly coupled to a particular load metric. Only the magnitude load balancing decisions, so that a load balancer can support any type of load metric, rather than just one type of metric. If a load balancer were load metric specific it would be costly to deploy load balancers for distributed applications that require balancing based on several metrics. For example, a separate load balancer would be needed to balance replicas based on various metrics such as CPU, I/O, memory, and network and battery power utilization. In addition, a load balancer must react to various replicas are balanced. For example, [19][15] when high load conditions occur, a replicas must be instructed to forward the client request back to the load balancer so subsequent request can be reassigned to a less loaded replicas.

Challenge 3:- *supporting modular load balancing strategies*

Since certain analysis techniques are not suitable for all use-cases, it may be useful to analyze a set of replicas loads in different ways depending on the situation. For example, to predict the future replicas loads it may be useful to analyze the history of loads for a given object group, thereby anticipating high load conditions. Conversely this level of analysis may be too costly in use-cases. E.g. if the duration of the analysis exceeds the time required to complete client request processing. Likewise, application developers may be interested in evaluating several alternative load balancing policies in which requiring a full recompilation or Relink cycle would unduly increase system development effort. A load balancing service cannot simply implement all possible load balancing strategies.

Challenge 4:- *Identifying objects uniquely*
When receiving information about the load in one replicas the load balancing service should determine the source of the load information efficiently and uniquely. CORBA[10] does not provide a lightweight mechanism to determine the source of a request. CORBA provides weak identity for objects, relying on the replica object reference to distinguish them would not be portable. So efficiently and portably determine is a bottleneck during the CORBA load balancing.

Challenge 5:- *Integration of the load balancing components*
All the components used in the CORBA architecture plays an important role at their own place and work in an independent manner. So all the components must be collaborate effectively to ensure that distributed system is properly load balanced. Direct interaction between some of the those components may complicate the implementation of distributed application, however since certain functionality may be exposed to a given component unnecessarily. So integration of all components in a well manner is another challenge in distributed systems.

III. Conclusions

This paper has presented a number of approaches for improving the performance of a distributed CORBA-based Service Control Point. Although distributed systems technologies can contribute greatly to this area by allowing processing requirements to be divided among a large number of less expensive processors, it is unwise to assume that increasing processing power or memory sizes of network processors ad infinitum will alone guarantee high performance. The solutions offered in this paper aim to increase the efficiency and cost effectiveness of resources with a view to making CORBA-based solutions more suitable for high performance, reliable systems required by telecommunications environments.

References

[1] S. Vinoski, *CORBA: Integrating diverse applications within distributed heterogeneous environments,* IEEE Communications, vol. 14, No. 2, Feb. 1997

[2] Common Object Request Broker Architecture (CORBA/IIOP) Specification, Version 3.0.2. The Object Management Group, December 2002. Available electronically at http://www.omg.org/technology/documents/corba_spec_catalog.htm

[3] Michi Henning, Advanced CORBA Programming With C++, Addison-Wesley, 1999

[4] Minimum CORBA, Version 1.0. The Object Management Group, August 2002. Available electronically at http://cgi.omg.org/docs/formal/02-08-01.pdf

[5] Cathy Hrustich, "CORBA For Real-Time, High Performance and Embedded Systems", Fourth IEEE International Symposium on Object- Oriented Real-Time Distributed Computing, 200. pp. 345-349

[6] Real-time CORBA, Version 1.1. The Object Management Group, August 2002. Available electronically at http://cgi.omg.org/docs/formal/02-08-02.pdf

[7] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and performance of Real-Time Object Request Brokers," Computer

[8] D. C. Schmidt, S. Mungee, S. Flores-Gaitan, and A. Gokhale, "Software Architectures for Reducing Priority Inversion and Non-determinism in Real-time Object Request Brokers," Journal of Real-time Systems, special issue on Real-time Computing in the Age of the Web and the Internet, To appear 2001.

[9] A. B. Arulanthu, C. O'Ryan, D. C. Schmidt,M. Kircher, and J. Parsons, "The Design and Performance of a Scalable ORB Architecture for CORBA Asynchronous Messaging," in Proceedings of the Middleware 2000 Conference, ACM/IFIP, Apr. 2000.

[10] B. Natarajan, A. Gokhale, D. C. Schmidt, and S. Yajnik, "DOORS: Towards High-performance Fault-Tolerant CORBA," in Proceedings of the 2nd International Symposium on Distributed Objects and Applications (DOA 2000), (Antwerp, Belgium), OMG, Sept. 2000.

[11] A. Gokhale and D. C. Schmidt, "Measuring the Performance of Communication Middleware on High-Speed Networks," in Proceedings of SIGCOMM '96, (Stanford, CA), pp. 306–317, ACM, August 1996.

[12] D. C. Schmidt and C. Cleeland, "Applying Patterns to Develop Extensible ORB Middleware," IEEE Communications Magazine, vol. 37, April 1999.

[13] I. Pyarali, C. O'Ryan, D. C. Schmidt, N. Wang, V. Kachroo, and A. Gokhale, "Using Principle Patterns to Optimize Real-time ORBs," Concurrency Magazine, vol. 8, no. 1, 2000.

[14] O. Othman, C. O'Ryan, and D. C. Schmidt, "The Design of an Adaptive CORBA Load Balancing Service," IEEE Distributed Systems Online, vol. 2, Apr. 2001.

[15] Object Management Group, Persistent State Service 2.0 Specification, OMG Document orbos/99-07-07 ed., July 1999.

[16] D. C. Schmidt, V. Kachroo, Y. Krishnamurthy, and F. Kuhns, "Applying QoS-enabled Distributed Object Computing Middleware to Next-generation Distributed Applications," IEEE Communications Magazine, vol. 20, Oct. 2000.

[17] L. R. Welch, B. A. Shirazi, B. Ravindran, and C. Bruggeman,"DeSiDeRaTa: QoS Management Technology for Dynamic, Scalable, Dependable Real-Time Systems," in IFACs 15th Symposium on Distributed Computer Control Systems (DCCS98), IFAC, 1998.

[18] IONA Technologies, "Orbix 2000." www.iona-iportal.com/suite/orbix2000.htm.
[19] L. Moser, P. Melliar-Smith, and P. Narasimhan, "A Fault Tolerance Framework for CORBA," in International Symposium on Fault Tolerant Computing, (Madison, WI), pp. 150–157, June 1999.
[20] Object Management Group, Fault Tolerant CORBA Specification, OMG Document orbos/99-12-08 ed., December 1999.
[21] BEA Systems, et al., CORBA Component Model Joint Revised Submission. Object Management Group, OMG Document