

Performance comparison of DCOM, CORBA and Web service

SeongKi Kim

School of Computer Science and Engineering
Seoul National University, 56-1 Sinlim, Kwanak
Seoul, Korea 151-742

SangYong Han

School of Computer Science and Engineering
Seoul National University, 56-1 Sinlim, Kwanak
Seoul, Korea 151-742

Abstract - The distributed object technologies are useful in many server situations where the high performance is required. However, the problems that many distributed object technologies such as DCOM, CORBA, and web service exist, but their comparison is insufficient cause a trouble to developers, administrators, practitioners who should choose one of them. This paper presents a performance guideline, advantages, and disadvantages to the prospective users making the users easily compare technologies, and select one of them.

Keywords: Distributed object, COM, CORBA, Web service, Distributed processing

1 Introduction

As internet users have increased, the high performance has been required in many server situations such as *Hyper Text Transfer Protocol (HTTP)* [1] and *File Transfer Protocol (FTP)* [2]. Although a single supercomputer approach occupied most of the markets and researches at an early stage, it had its own limitations in the aspects of a cost, reliability, and scalability. The limitations made parallel processing approaches such as a *Cluster* [3], *Beowulf* [4][5], or a distributed processing approach more important. In addition, the fast growth of a network performance compared with the slow growth of a CPU performance has accelerated the multiple computer approaches [6][7][8]. As an example of showing this trend, the occupied ratio of clusters in a supercomputer increases every year [9]. A distributed processing is also widely used with a parallel processing approach at these days, especially in multi-tier environments where separates a business logic from a data access logic, and where has important legacy systems.

Among these distributed processing technologies, *Remote Procedure Call (RPC)* [10] and *Remote Method Invocation (RMI)* [11] exist as early models. However, as the object oriented paradigms flourished, these early models have evolved into distributed object technologies such as *Distributed Component Object Model (DCOM)* [12], *Common Object Request Broker Architecture (CORBA)* [13], and eventually a web service [14] to provide services at the web environments.

However, developers, administrators, and practitioners have a lot of difficulties in choosing their own distributed object technology, sometimes want to change it later with startled by the unexpected problems because it was chosen without a guideline or comparison. This paper describes the advantages and disadvantages of each technology, presents a guideline to developers, vendors, and practitioners in the aspect of a performance making them expect the performance when using one of distributed object technologies.

This paper is organized as follows. Section 2 shows the various related works by many eminent researchers such as DCOM, CORBA, and web service. Section 3 describes their measured performances. Section 4 concludes.

2 Related Works

This section describes the distributed object technologies such as DCOM [12][15], CORBA [13], and web service [14][16].

2.1 DCOM

DCOM is a distributed object technology that was developed by *Microsoft*, and extended COM to fit into network environments. Fig. 1 shows the COM/DCOM architecture.

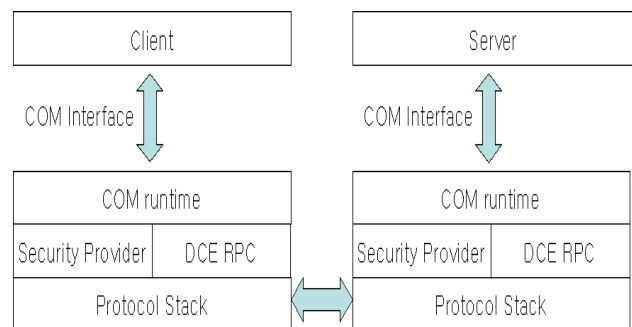


Fig. 1. COM/DCOM architecture

Fig. 1 shows the COM/DCOM architecture. A client that wants to access a server object calls a COM runtime through COM interfaces, which checks the client's permissions by invoking a security provider. If the client has the appropriate permissions, the COM runtime invokes the *Distributed Computing Environment Remote Procedure Call (DCE RPC)* that uses the underlying protocol stack as one of communication methods. Before invoking a DCE RPC, a client side COM runtime performs a *marshalling* through a *proxy* that converts parameters into the form that could be transferred over the designated protocol. The protocol stack on the server side receives the request, delivers it to the COM runtime, which performs an *unmarshalling* through a *stub* that converts network packets into parameters. The COM runtime activates the server object that calls the object method and processes the request.

A server object has the 3 types such as an *inproc*, a *local*, and a *service* type. An *inproc* type takes the form of a *Dynamic Linking Library (DLL)*. An *inproc* type is the fastest type because it is loaded directly into a client process when activated. A *local* type takes an executable form. When activated, it is executed as a separate process. Lastly, a *service* type is also similar to the *local* type in that both of them take executable forms, but different from the *local* type in that the *service* type is always resident in a memory, and receives a request from a client.

In order to export the methods of a server object, a server object should describe itself by using an *Interface Description Language (IDL)*. For methods that are commonly used in the COM, COM previously creates a set of common interfaces such as *IUnknown*, *IDispatch*, and *IClassFactory*.

Although Windows operating system incorporates it, both COM and DCOM are supported only by limited operating systems, and they can't be used in most of web environments due to protection policies.

2.2 CORBA

CORBA is a distributed object technology that was specified by *Object Management Group (OMG)* and implemented by many vendors such as *Borland*, and *IONA*. Fig. 2 shows the CORBA architecture.

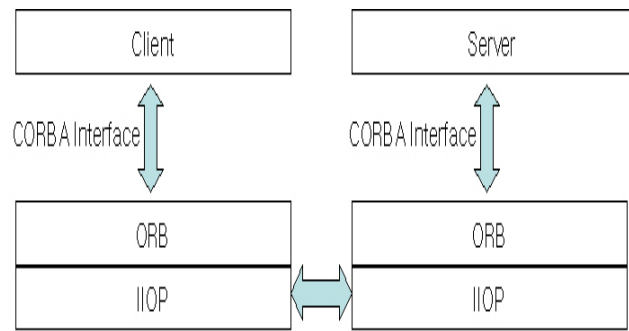


Fig.2. CORBA architecture

A client that wants to access a server object should first bind it through a CORBA interface. The client makes a bind request to a local *Object Request Broker (ORB)* through CORBA interfaces. If the object exists at the local machine, the ORB activates locally the object that processes the request. Otherwise, the ORB makes a request to the other ORBs connected by the network using the *Internet Inter ORB Protocol (IIOP)* over TCP/IP. After binding the appropriate object, the client side ORB sends the requested method call to the server side ORB. Before sending and receiving parameters, a *stub* performs marshalling and a *skeleton* performs unmarshalling at the CORBA.

CORBA has the 2 server types such as a *Basic Object Adapter (BOA)* and a *Portable Object Adapter (POA)*. A BOA type was the early model, and its ambiguity caused many different implementations at an early stage. A POA type was developed to solve this incompatibility problem, and came to be a standard. A CORBA also has the IDL that is similar to the COM IDL.

Although CORBA is supported by various operating systems, its users should also choose their own CORBA vendors because various vendors implementing CORBA specification exist, and they can't be used in most of web environments due to protection policies.

2.3 Web service

Web service is developed in order to distribute an object and serve it to various users in the web environments. It can be also used in the server situations while solving the web-scalability problem of the other distributed object technologies. Fig. 3 demonstrates the web service architecture.

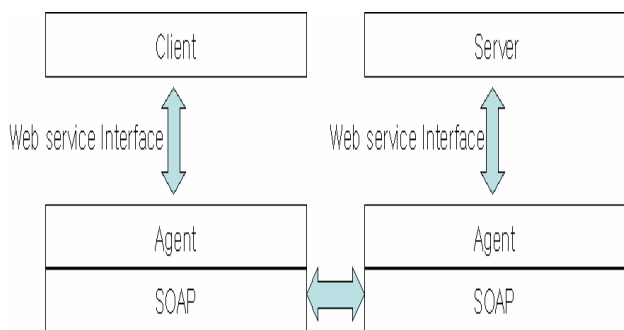


Fig. 3. Web service architecture

A client that wants to be serviced should first find the supported services from the pre-existing *registry* before compiling a code. After finding its services through searching, the client gains the *Web Service Description Language (WSDL)* that a server previously registers. From the WSDL, the client knows the service provider location and the parameters to the found method. After the client binds the described service during the compile time, it calls the local agent whenever the client invokes a method call, and the local agent delivers it to the server side agent through *Simple Object Access Protocol (SOAP)* over HTTP, FTP, SMTP, IIOP, and TCP during the runtime. The server side agent activates the appropriate object, and delivers the calls to the object.

All of the communication methods such as a WSDL, and SOAP exploit the *eXtensible Markup Language (XML)* [17]. WSDL is an XML describing the web service. SOAP is an XML describing the called method, its parameters, and its return value, and can be delivered over the HTTP. For this reason, web service can be used in any web environments without having to worry about the firewall.

Although web service could be used regardless of a firewall and is supported by various operating systems and languages, web service has the serious overhead that is caused by XML-creation.

3 Performance

This section describes the measured performance at the local machine and networked machines.

3.1 Local machine

We measured the performance on a machine having the *Pentium 4 1.72 Ghz*, and *512MB RAM* as a hardware. We used the *Windows XP*, *Visibroker 6.5*, *Visual Studio .NET 2003*, and *IIS 5.1* as softwares. As an implementation language, we chose commonly C++ in order to avoid the overheads according to the language difference. We repeated 1000 invocations of a method with various parameters. Table 1 shows the average of 10 repetitions of the calls.

Table. 1. The measured performance on a local machine

millisecond	Int	Char	Byte	Char array	Struct
Inproc	0	0	0	0	1.6
Local	550	712.5	551.5	553	534.4
Service	536	534.3	723.4	579.6	546.8
BOA	617.2	553	561.1	576.4	545.4
POA	495.6	526.4	523.3	629.7	498.6
WS	15164	11700	11344	13736	11198

An inproc server type of COM is the fastest among all of tests. At a local machine, the inproc server type is directly loaded into the executing process, thus this type has no overhead of *Inter Process Communication (IPC)* and is operated as a general method call. A web service type has the most serious overheads as estimated. In the case of a web service, the client side agent should perform the marshalling that converts both the method and its parameters into the SOAP message and deliver the message to the bound web server. The server side agent should perform the unmarshalling that converts a SOAP message into both the method and parameters, activate the server object that processes the request, and perform the marshalling for returning a value after a server object processes the request. The client should perform the unmarshalling to know the return value. The marshalling process of a web service call is complicated in that it should create an XML message. Another factor of the slow web service is that SOAP has one more network layer than DCOM, and CORBA. Although the underlying protocols of DCOM are RPC, TCP/IP, and those of CORBA are IIOP, TCP/IP, those of web service are generally SOAP, HTTP, TCP/IP. These factors conjunctively make the overheads of a web service significant. When it comes to the types, there is little difference among the types within the same technology when considering Table 1. This means that the parameter-passing overheads are trivial. Table 1 also shows that the overheads difference between COM and CORBA is little on a machine.

3.2 Networked machines

We measured the performance with another machine having the *Pentium 3 450 Mhz*, *320MB RAM*, *Windows 2000*, *Visibroker 6.5*, and *Visual Studio .NET 2003* as a client machine. We also repeated the 1000 invocations of a

method with various parameters. Table 2 shows the average of 10 repetitions.

Table.2. The measured performance on the networked machine

millisec onds	Int	Char	Byte	Char array	Struct
Inproc	510.3	593	632.2	598.2	612.3
Local	1042.8	1095.4	1678.2	1089.1	1203.5
Service	1132.3	1034.2	1328.9	1298.3	1132.4
BOA	2051	1593.9	2080	2073.9	2086.1
POA	1396.1	1458.1	1415	1458.2	1418.2
WS	14500	11403	11352	13156	11469

In Table 2, the inproc server type is also the fastest among all of tests. DCOM takes approximately 2 times more than Table 1, and CORBA takes approximately 3 times more than Table 1. At the networked machines, the technologies additionally have the network overheads and the security checking overheads. In the case of a web service, the overheads are similar to those of a local machine because the calling procedures are the same as those of a local machine, and additionally include only the network latency. The type difference is also trivial in a networked case.

4 Conclusion

As the internet users increase, the distributed object technology is placed an emphasis on. The distributed object technology has its merits that the servers can be logically separated and used even in the case of having an important legacy system. Among these distributed object technologies, DCOM, CORBA, and web service exist. However, choosing one of them is complex and requires much consideration because the comparison data lack.

This paper provides its prospective users with a guideline about the elapsed time, makes the users expect their overheads when one of them is chosen, and compares their own advantages and disadvantages. Through our experiments, we found that the inproc type of COM and DCOM is the fastest among distributed object technologies at a local machine and between the networked machines, the invocation among the networked machines takes 2 or 3 times more than the invocation on a local machine, and the

parameter passing overheads are trivial compared with the network overheads. We thought that to the best of our knowledge, this paper is valuable because this paper is the first comparison among DCOM, CORBA, and web service.

4 References

- [1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.1, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>.
- [2] J. Postel, J. Reynolds, FILE TRANSFER PROTOCOL (FTP), October 1985, <http://www.ietf.org/rfc/rfc0959.txt>.
- [3] Task Force on Cluster Computing home page, <http://www.clustercomputing.org>.
- [4] A.J. van der Steen, An evaluation of some beowulf clusters, Technical report, Utrecht University, Department of Computational Physics, December 2000.
- [5] T.L. Sterling, J. Salmon, D.J. Becker, D.F. Savaresse, How to Build a Beowulf, Book, Boston, 1999.
- [6] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, Wen-King Su, Myrinet - a gigabit-per-second local area network, IEEE Micro, Vol. 15, No. 1, pp. 29-36, January 1995.
- [7] T. Shanley, Infiniband Network Architecture, Book, November 2002.
- [8] D.B. Gustavson, Q. Li, Local-area multiprocessor: the scalable coherent interface, Technical report, Santa Clara University, Department of Computer Engineering, 1995.
- [9] TOP500 Supercomputer sites, <http://www.top500.org>.
- [10] R. Srinivasan, RPC: Remote Procedure Call Protocol Specification Version 2, August 1995.
- [11] Sun Microsystems, Java RMI specification, February 1997.
- [12] Microsoft, The Component Object Model specification, October 1995.
- [13] OMG, Common Object Request Broker Architecture, March 2004.
- [14] W3C, Web Service Architecture, <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>, February 2004.

[15] R. Grimes, Professional DCOM Programming, Book, Wrox press, June 1997.

[16] W3C, Web Services definitions, <http://www.w3.org/2002/ws>.

[17] Francois Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Extensible Markup Language (XML) 1.0 (Third Edition), 4 February 2004, <http://web3.w3.org/TR/2004/REC-xml-20040204/>.