Performance Comparison of ICE, HORB, CORBA and Dot NET Remoting Middleware Technologies

Sumair Khan
Department of Computer science,
COMSATS Institute of Information Technology,
Abbottabad, Pakistan

Kalim Qureshi
Information Science Department,
Kuwait University,
State of Kuwait

Haroon Rashid
Director of Campuses,
COMSATS Institute of Information Technology,
Islamabad, Pakistan

ABSTRACT

Distributed computing systems are designed to solve computationally intensive problems with the help of convergence of computing resources scattered across the network. Distributed computing object middleware technologies have bring revolutionary concepts in the world of distributed computing and also made the building of distributed computing applications more efficient and nearer to real world. But the selection of most efficient distributed computing object middleware technology on the basis of different performance metrics is an important research issue.

In this paper we are presenting the performance evaluation and comparison of distributed computing object middleware technologies which include Common Object Request Broker Architecture (CORBA), Internet Communication Engine (ICE), HORB, and TCP based Dot NET Remoting. Because these distributed computing object middleware technologies have not been evaluated and compared collectively on the basis of performance metrics which include overhead generation and round trip latency. The results that we have gathered showed that ICE is showing better performance in terms of overhead generation. And HORB has showed reduced round trip latency as compared to other middleware's.

Keywords

Performance Evaluation, Distributed Computing Object Middleware technology, CORBA, HORB, ICE, Dot NET Remoting

1. INTRODUCTION

Nowadays, due to the distributed objects, the possibility to develop distributed applications with components located on any machine in a network is a reality with the help of distributed computing object middleware technologies. Day by day demand of performance efficient distributed applications is growing to accomplish different critical tasks related with distributed computing. To achieve this efficiency in distributed the applications depends upon the selection of appropriate distributed computing middleware technology which has better performance as compared to other distributed computing middleware technologies available.. Previously researchers have published done the research work referring to choosing the most efficient distributed computing object middleware technology from number of technologies to build the performance efficient distributed applications. But in this research work we have selected the four distributed computing object middleware technologies which include Common Object Request Broker Architecture (CORBA)[1] to implement CORBA we have selected its Java IDL[2] implementation of CORBA because its feature of open source make it prefer candidate implementation as compared to other implementations, Internet Communication Engine (ICE)[3] it's the product of ZeroC, HORB[4] developed by AIST, and Dot NET Remoting[6] developed by Microsoft. These selected technologies are offering the advance features required by today's applications. These selected distributed computing object middleware technologies have not been compared collectively on the basis of different performance metrics which include overhead generation and round trip latency.

Most of research work done so far is based on common distributed computing object middleware technologies which include CORBA, RMI, and DCOM. In [6] author has compared the performance of CORBA, RMI and DCOM on the basis of speed of the message transferred between the applications. In [7] again authors have compared the performance of CORBA, DCOM and RMI but using the different performance metrics. In [8] authors have done the performance analysis, comparison and optimization of the distributed object middleware technologies based on Java which include RMI and RMI-IIOP. In [9] an extensive research has been performed on the performance efficiency of RMI and then it is compared with other distributed object middleware's.

In our research work selected distributed object middleware technologies have not been compared collectively on the basis of different performance metrics This research will give us a clear picture of performance of selected middleware technologies, so that selection of the most appropriate middleware technology with respect to different performance metrics will become easy for developers to develop efficient distributed applications, because today' distributed applications requires low overhead generation generated due to computation and need low latency to accomplish the required performance level. To accomplish the objective of this research work, number of performance evaluation tests has been performed and there results have been discussed. Rest of the paper is organized as follows: in next section we will discuss the selected Distributed Computing Object Middleware Technologies, after that we will describe the experimental setup environment, then we will discuss the results in Performance Evaluation and Comparison section and last will be the conclusion section.

2. DISTRIBUTED COMPUTING OBJECT MIDDLEWARE TECHNOLOGIES

In this section distributed object middleware technologies which we have selected for the performance evaluation and comparison will be discussed.

2.1 ICE (Internet Communications Engine)

ICE (Internet Communications Engine) has emerged as a new object-oriented middleware having advance features than traditional object-oriented such as DCOM and CORBA. With the evolution of Object-oriented middleware, distributed applications also entered into new paradigm of computing. ICE middleware platform care important functionalities provided by any middleware, such as marshaling and unmarshaling mapping logical object addresses to physical transport endpoints, changing the representation of data according to the native machine architecture of client and server, and automatically starting servers on demand. ICE architecture consists of different components which include ICE core, slice, proxy code, skeleton code, object adapter and ICE Protocol[3].

2.2 Dot NET Remoting

Dot NET Remoting is based on dot NET framework that enables us to build distributed applications which can solve the distributed application problems. Dot NET remoting provides an infrastructure for distributed objects through which we build applications that use objects to communicate across the network. Dot Net Remoting architecture consists of components which include client object, server object, proxy object, channels and formatters [5].

2.3 CORBA

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture developed by the Object Management Group (OMG). CORBA is a distributed object-oriented client/server platform used to build the object based distributed applications. CORBA includes object-oriented Remote Procedure Call (RPC) mechanism that provides the basic remote procedure call services. Primary components in the CORBA architecture include Object Request Broker (ORB), IDL stubs, ORB interface and object adapter [1].

2.4 HORB

HORB is light weight distributed computing middleware that extends the java functionality into the network computing to facilitate in developing java based distributed applications. The programs written in HORB can be run on any type of operating systems. This feature is inherited from Java on which HORB is based on programs written in Java runs greatly on many machines without recompilation. Using the HORB interoperability feature, we can run a HORB program spreading over many different kind's of machines. HORB architecture consists of several components which include proxy object, skeleton object, Object Request broker (ORB) [4].

3. EXPERIMENTAL SETUP

To perform the performance evaluation of middleware technologies, we setup the client-server environment. The test environment consists of two computers one acting as client server and the other acting as server and both are connected via a 100Mbit network connection. Windows XP was chosen as machine's operating system because it can host all the software's required for the experimentation. The software's that were used in experimentation included Sun JDK 1.5 that was used to implement CORBA, HORB and ICE implementations, Dot NET framework Platform was used to test the TCP based Dot Net Remoting, and last profiler were also used to get the experimental results relevant to overhead generation which included ethereal [11], hpjmeter profiler [12] and CLR Profiler [13].

4. PERFORMANCE EVALUATION AND COMPARISON

In this section we will report the measured results that we have gathered during our tests performed on our selected middleware technologies: ICE, CORBA, HORB and Dot Net Remoting.

4.1 Testing Methodology

The tests we have performed on the middleware's were divided into two main categories. First one was Overhead generation test category and second one was Round Trip Latency test category. In each test category we have performed two tests; first was primitive data type test (which was based on single data value of specific data type) and the second was the Array based test (which was based on bulk of data was used in tests).

In Round trip latency primitive data type test we also checked the round trip latency of the middleware's under the scalability factor. Means how the round trip latency is affected by increasing number of clients accessing the server machine simultaneously.

In the next sections we will discuss the results of the tests that we have conducted.

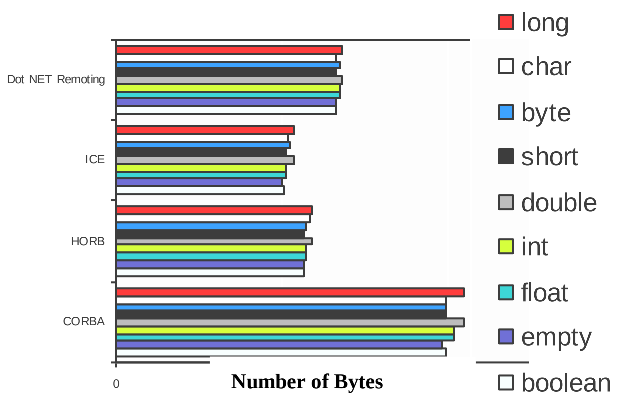
4.2 Overhead Tests

Basic aim of these tests to check how much extra network traffic is generated by the middleware technologies to transmit the actual data over the network. As we have stated earlier that we have performed two tests on each middleware technology, one was primitive data type test and the second one was array based test. Now we will discuss the results of each test one by one.

4.2.1 Primitive Data Type Test

This test was performed to know the behavior middleware technologies in terms of overhead generation when different method were executed on the remote machine with the help of data values of different data type passed to the remote methods. In Figure 1 we are presenting the results gathered from overhead measurements for primitive data types and empty method call. we can observe the overhead results of all primitive data types in different distributed computing object middleware technologies are generating overhead according to there size in bits . Long and double primitive data types are producing higher overhead as compared to other data types in all distributed computing object middleware technologies; because long and double have 64-bit representation in all distributed computing object middleware technologies. If we compare overall behavior of our selected middleware technologies we can observe that Java IDL is producing highest overhead as compared to other middleware technologies. The most important reason behind this behavior is that the IIOP Protocol of CORBA uses Common Data Representation (CDR). The CDR padding rules unfortunately generates higher level of network overhead.

Figure 1. Overhead generation (Primitive Data Types)



On the other hand ICE is showing the better results with respect to network overhead generation and it is generating less than 50% of the network overhead as compared to network overhead generated by CORBA. The reason behind this behavior is that encoding rules used by ICE are very compact, which in the result reduce the overhead generation. As we can observe from the results that HORB is showing better results than Dot NET Remoting because it doesn't use any type of connection multiplexing because connection multiplexing produces excessive data copying which in result reduces network overhead but due to encoding rules of HORB behaves worst as compared to ICE due to which it is showing not the better results as compared to ICE.

4.2.2 Array based Test

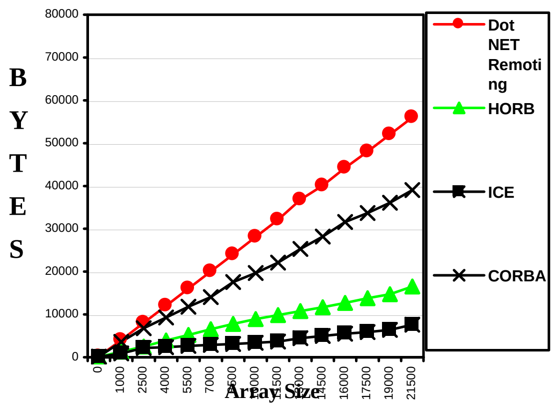
For the array based test, the premises are the same as for the primitive data type's test, with the exception that the client applications are set to different array sizes for the selected primitive data type. We have selected Double data type for Array based test because Primitive Data Type Test has shown us that double primitive data type is showing worst results in all four middleware technologies as shown in figure 1.

In this test we have evaluated the performance of distributed computing object middleware technologies on the basis of different array sizes of double data type. In figure 2 we can observe the behavior of middleware technologies on the basis of array passing.

If we compare the measured results in figure 1(Primitive data type test) and figure 2(Array based test) we can observe that Dot NET Remoting has taken the position of CORBA in terms of overhead generation and as graph shows Dot Net Remoting is generating four times more overhead as compared to ICE which is showing the best results. Reason behind this behavior of Dot NET Remoting is its inefficiency of handling large size data. Therefore as we increase the size of array the Dot NET Remoting overhead generation also increases as shown in figure 2.

ICE is showing the best performance than all other distributed computing object middleware technologies the reason behind this behavior is that ICE is using more compact encoding Technique as compared to all other middleware technologies.

Figure 2. Overhead generation (different Array sizes)



4.3 Round Trip Latency Tests

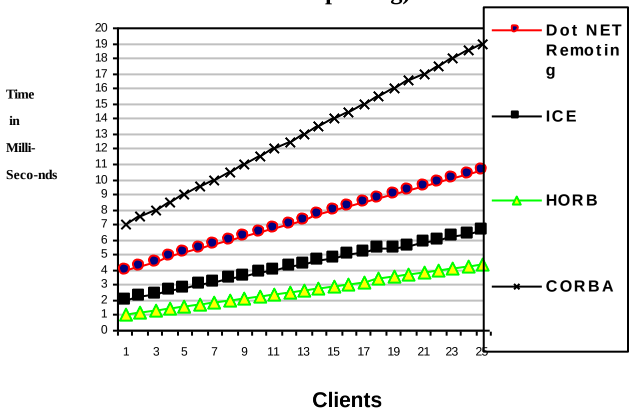
Main Motivation behind these tests to check the round trip latency in terms of number of milliseconds per call required by the middleware technology for the complete execution of the method call. In Primitive Data Type Test we have also checked the round trip latency on the basis increasing number of Clients, means how the distributed computing object middleware technologies give the round trip latency performance under the load of more than one client. Scalability is also an important performance metric to check the performance of middleware technologies.

4.3.1 Primitive Data Type Test

In this test we have send the primitive data type value which was single double value to check the round trip latency times of selected middleware technologies. We have tested the scalability factor of middleware's with the help of increasing number of client requests. We can observe from the figure 3 that HORB is showing the best performance as compared to all other distributed computing object middleware technologies.

As we can observe from the graph in figure 3 that as the number of clients are increasing the performance difference between HORB and other middleware technologies is also increasing. HORB is producing four times less round trip latency as compared to CORBA which is showing the worst results. Reason behind this behavior of HORB middleware technology is that HORB use the system object serializer which includes ObjectOutputStream and ObjectInputStream. And ObjectOutputStream does its own buffering.

Figure 3: Round Trip Latency (single data value passing)

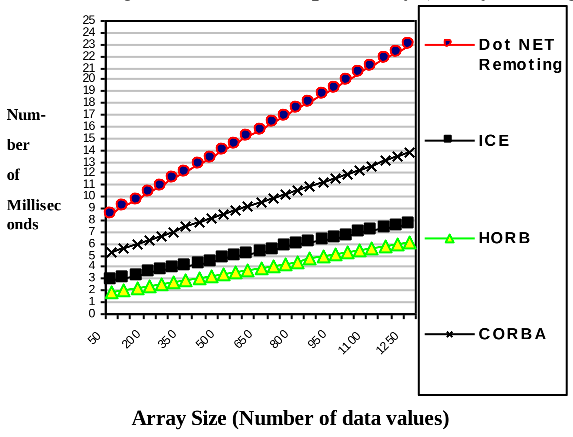


And in result serialization process take less time as compared to serialization time taken by other middleware's. On the other hand Java CORBA is showing worst performance in terms of Number of milliseconds per call because CORBA uses padding in the process of CDR, due to which serialization process slows down. So in result round trip latency of CORBA becomes worst as compared to all other distributed computing object middleware's. ICE is consuming minimum Number of millisecond per call as compare to Dot NET remoting because the results we have gathered showed that ICE is consuming less Time in the process of serialization as compared to Dot NET remoting. This means that ICE serialization process is faster than Dot NET Remoting serialization process.

4.3.2 Array based Test

Basic aim of this test is to check the performance in terms number of round trip latency when user wants to send bulk of data in terms of array passing. In this test we have used Array of different sizes of Double data type. From figure 4 we can clearly observe that Dot NET Remoting is showing the worst performance in handling of large size data because the serialization process is taking more time according to the results we have gathered. It has approximately five times worst round trip latency as compared to HORB. On the other hand HORB is producing the best results due to its serialization process efficiency. ICE is showing the better results as compare to CORBA and Dot NET remoting.

Figure 4: Round Trip Latency (Array Passing)



5. CONCLUSION

In this paper we have done a detailed performance evaluation and comparison of important distributed computing object middleware technologies which include ICE, HORB, Dot NET Remoting and CORBA on the basis of performance metrics which includes overhead generation and round trip latency. These performance metrics have been measured under different test scenarios.

Overhead tests showed that the overhead bottlenecks appeared in the distributed computing object middleware technologies were due to many factors which includes excessive data copying, less compact encoding and complex encoding rules. ICE showed the better results as compared to other middleware technologies for the overhead generation tests for any data type of any data size. In the case of round trip latency tests we have found that HORB is producing low round trip latency because it uses external serializer for the process of serialization. The results which we have gathered with the help of profiler showed that HORB serialization Process takes less time as compared to ICE serialization Process.

From the results we can suggest that for the development of distributed applications ICE is most efficient middleware as compared to other middleware technologies in terms of overhead generation. And for time critical distributed applications, HORB is the best selection for building time critical distributed applications.

6. REFERNCES

[1] The Common Object Request Broker: Architecture and Specification, Revision 2.3: June1999; www.omg.org/docs/formal/99-10-07.pdf

[2] http://www.cs.rug.nl/~gert/docs/java/guide/idl/jidlUsingCORBA.html

[3] Distributed Programming with Ice, Michi Henning, Mark Spruiell; www.zeroc.com/**ice**.html

[4] http://horb.aist.go.jp/horb/doc/guide

[5] Microsoft® .NET Remoting**,** Scott McLean, James Naftel, Kim Williams, Microsoft Press.

[6] Florian Mircea Boain Aan Rares, RMI VERSUS CORBA: A MessageTransfer SPeed Comparison, Stidia Univ. Babes{BOLYAI, INFORMATICA, Volume XLIX, Number 1, 2004

[7] A Bracho, A. Matteo, Ch. Metzner, A Tanxonomy For Comparing Distributed Object Technologies**;** CLEI Electronic Journal 2 (2), 1999.

[8] Matjaz B. Juric, Ivan Rozman, Alan P. Stevens, Marjan Hericko, Simon Nash, Java 2 Distributed Object Models Performance Analysis, Comparison and Optimization;2000 IEEE

[9] Sanjay P. Ahuja, Renato Quintao, "Performance Evaluation of Java RMI: A Distributed Object Architecture for Internet Based Applications", 8th International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems, August, 2000.