CORBA BASED CONTROL SYSTEM WITH RTOS ON VME/CPCI

Toshiya Tanabe, Toshikatsu Masuoka, Jun-ichi Ohnishi, Yasushi Watanabe
and Masayuki Kase, RIKEN, Saitama, JAPAN

Abstract

R&D work on CORBA based control system for RIKEN-RI Beam Factory (RIBF) [1] has been undertaken at RIKEN. Various tests using RTOSes such as VxWorks and pSOS running on VME or CPCI and corresponding ORBs have been conducted to evaluate the performance. This paper describes the methods and results of the test and our future plan of RIBF control system.

1 INTRODUCTION

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture for an open software bus, which allows object components created by different OSes and languages to interoperate. It would make integration with legacy controls easier, which is well suited for RIBF project as it is an expansion of existing facilities. In recent years, Java/CORBA applications for business environment have been flourishing. However, the progress with CORBA for embedded computing is slow. We have therefore concluded that it is premature to completely replace VME/CPCI on a RTOS with PCs for our project. Therefore, as a part of R&D work for RIKEN RIBF control system, we have conducted various tests of ORB for RTOSes on VME/CPCI.

2 CORBA TEST ON VME/CPCI

2.1 General Description

The main purposes of our R&D are:

- To establish CORBA based communications among different GUIs (created in C++ and Java) and VME/CPCI running different RTOSes.
- To estimate the total operational overhead due to the use of CORBA.
- To confirm interoperability through Internet Inter-ORB Protocol (IIOP) among different ORBs.
- To understand the deficiencies, if any.

Besides the interoperability test, the ORB used for all components is chosen to be VisiBroker [2] due to the availability for both VxWorks [3] and pSOSystem [4] as well as its ubiquitous availability through Netscape browser. Visual C++ 6.0 is used for C/C++ applications and Visual Cafe for Enterprise Edition [5] for Java. Figure 1 shows the system configuration for this R&D.

2.2 From Socket Programming to CORBA

A few GUI applications are made in C++ which utilize conventional socket functions to communicate with VxWorks running on a PowerPC604 based VME (Motorola MVME 2600). There are a variety of VME I/O boards with a VxWorks driver that we have installed and tested their functions. Then some sources for each layer are modified and compiled with links to CORBA libraries. By using CORBA, one can discard routines for socket functions which require detailed information on the hardware characteristics on both sending and receiving sides. The client which may not know the details of the server simply invokes the server object to complete a two-way communication. Table 1 shows a comparison between two schemes.

Fig. 1: CORBA test system configuration

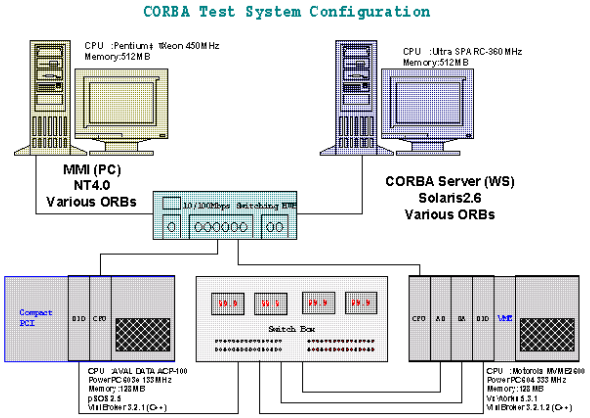

Table 1: Comparison on programming aspects between socket communication and CORBA.

	Socket	CORBA
Program Making	Define exact data formats for send / receive	As if one defined a stand-alone function
Endian	Endian problems for binary transmission	CORBA absorbs endians
Debug-ging	Confirm the formats on both sides	As if one checked function calls
Program Change	Modification required on both sides	Treated as separate objects

2.3 DII and Naming Service

Under normal circumstances for accelerator controls, the interface information is known and static. Therefore, Static Invocation Interface (SII) is adequate. Dynamic Invocation Interface (DII) could be used in case the types

of parameters used in the programs are unknown. The program of the client that uses DII will be more complex because treatment must be described to look up the argument of the server, return value, and so on dynamically. As far as RIBF controls are concerned, DII is deemed unnecessary. Naming service (NS) is useful in case one kind of ORB is used for the entire system. For comparison, the response times for various I/O boards have been measured for these three different methods (SII, DII and NS). Test routines are in the following: (a) Server returns I/O values on the request of a client. (b) Client invokes the object 100 times and the elapsed time is measured. This routine is repeated 10 times to get better statistics.

2.4 Interoperation Between Different ORBs

Besides VisiBroker, a free ORB (for non-commercial use only) such as ORBacus[6], is used to evaluate the interoperability among different ORBs. Interoperable naming service is not available until CORBA 3.0 [7] so one of the ways to get the object reference across different ORBs is as follows: (1) Interoperable Object Reference (IOR) is changed into the string, and a server-side writes it in the file. (2) A client-side reads the file created on the server-side, and returns a string to IOR. A file is delivered with FTP, NFS, and so on. The example codes for both the server and the client are given below.

Client Program(CORBA Product:ORBacus)

```
#define  LOOP 10000
void main(int argc, char *argv[])
   {
CORBA_ORB_var orb = CORBA_ORB_init(argc, argv);
      const char* refFile = "ior.ref";
      ifstream in(refFile);
      char s[1000];
      float _rnumber;
      in >> s;
//String is being changed into the IOR.
CORBA_Object_var obj = orb -> string_to_object(s);
      Test_var test = Test::_narrow(obj);
   for(int i=0;i<LOOP;i++){
      _rnumber = test -> rnumber(); }
      cout << "last rnumber = " << _rnumber << endl;
      cout << "Hit any key for " ;
      cin >> s;
   }
```

Server Program(CORBA Product:VisiBroker)

```
void main(int argc, char *argv[])
   {
CORBA_ORB_var orb = CORBA_ORB_init(argc, argv);
CORBA_BOA_var boa = orb -> BOA_init(argc, argv);
      Test_var p = new Test_impl;
//IOR is being changed into the String.
CORBA_String_var s = orb -> object_to_string(p);
      const char* refFile = "ior.ref";
      ofstream out(refFile);
      out << s << endl;
      out.close();
      boa->
 impl_is_ready(CORBA_ImplementationDef::_nil());
}
Test_impl::Test_impl()
{
}
float Test_impl::rnumber()
{
   float _rnumber;
  // Set a random number between 0 to 10000
   _rnumber = abs(rand()) % 100000 / 100.0;
   return _rnumber;
}
```

Turn around times (TATs) for different configurations have been measured and compared. The method is as follows: (a) Server returns a random number (float) when it is called upon by a client. (b) Client invokes the object 10000 times and the elapsed time is measured. This routine is repeated 10 times to get better statistics.

2.5 pSOSystem on CPCI with PowerPC 603e

One of our goals is to establish heterogeneous environment in accelerator controls. Therefore, a RTOS other than VxWorks should be tested. pSOSystem has been chosen this time due to the availability of ORB, especially VisiBroker. A compact PCI CPU board (AVAL DATA ACP-100) [8] is used as a main controller. CPCI-DIO board from the same manufacturer [9] as the corresponding VME board is installed for comparison. Unfortunately, numerous unforeseen problems prevented us from completing this test in time for publication.

3 TEST RESULTS

3.1 Measurement of Response times for I/O boards

In this test, for the client, Windows-NT4.0 is used for the OS and Visual C++6.0 is for the compiler. For the server, VxWorks 5.3.1 running on PowerPC604 with C++ as a compiler are employed. VisiBroker is used for both the client and server. Interface Definition Languages (Idls) used for each board is in the following:

```
/*      for AD Board                */
interface Ad_Board {
        short      Ad( out string Str_Ad );  };
/*      for DA Board                */
interface Da_Board {
        short      Da( in string Str_Da );  };
/*      for DIO Board     */
interface Dio_Board {
        short      Di( out string Str_Di );
        short      Do( in string Str_Do );  };
```

Table 2 shows the measurement times in msec for different methods. It seems that rather large fluctuation in response for socket communication is due to the use of an event-driven socket library (CAsynchSocket). Some events not related to socket communications might have occurred during transmissions. As for DII cases, this particular test somewhat sacrifice the generality. The server idls are given a priori so that the time which DII searches for the interfaces is shortened.

Table 2: Response Times for Various I/O Boards [msec]

Client \ Server		VisiBroker (C++)			ORBacus(C++)	
		Sun	NT	VxWorks	Sun	NT
VisiBroker (C++)	Sun	519*[0.0819]	428 [0.191]	735 [0.0481]	380 [0.0525]	455 [0.178]
	NT	492 [0.191]	317*[0.00876]	664 [0.0154]	413 [0.0838]	321 [0.0115]
ORBacus (C++)	Sun	466 [0.103]	417 [0.269]	768 [0.0467]	351 [0.158]	460 [0.422]
	NT	497 [0.0886]	327 [0.0158]	727[0.00422]	406 [0.179]	328 [0.00738]
ORBacus (Java)	Sun	1340 [0.129]	1237 [0.102]	1635[0.0706]	1243 [0.0953]	1256 [0.0644]
	NT					

3.2 Interoperation

The results of interoperations among different ORBs and OSes are shown in Table 3. The numbers with an asterisk indicate that a VisiBroker function utilizing interprocess communication is used. So far only ORBacus for java has been tested for java/CORBA implementation. JDK 1.2 has been used for the evaluation. It suggests that java client for this particular ORB is approximately 2.5 ~3.5 times slower than C++ implementation for the same task. It is yet not certain that variations in response among different OSs are due to those in hardware performance or in software design. It appears that VxWorks as a server gives smaller fluctuations in response than other OSs as expected for RTOS.

Table 3: Response times Matrix for Various ORBs and OS's (Average [Std. Dev.] in μsec). Asterisked numbers using IPC are 446 and 177μsec, respectively.

Client \ Serve		VisiBroker (C++)			ORBacus(C++)	
		Sun	NT	VxWorks	Sun	NT
VisiBroker (C++)	Sun	519*[0.0819	428 [0.191]	735 [0.0481]	380 [0.0525]	455 [0.178]
	NT	492 [0.191]	317*[0.00876	664 [0.0154]	413 [0.0838]	321 [0.0115]
ORBacus (C++)	Sun	466 [0.103]	417 [0.269]	768 [0.0467]	351 [0.158]	460 [0.422]
	NT	497 [0.0886]	327 [0.0158]	727[0.00422	406 [0.179]	328 [0.00738
ORBacus (Java)	Sun	1340 [0.129]	1237 [0.102]	1635[0.0706	1243 [0.0953	1256 [0.0644
	NT					

For comparison, the times using socket programming are shown in Table 4. This time, instead of the event driven socket library (CAsyncSocket), winsock.dll is used to measure genuine elapsed time for socket communication.

Table 4: Response Times for socket communication

Client \ Server		Socket		
		Sun	NT	VxWorks
Socket	Sun	110 [0.0242]	180 [0.0389]	185 [0.0094
	NT	180 [0.0283]	121 [0.0135]	165 [0.0080
	VxWorks			

4 FUTURE PLANS

It is now preferable to create GUIs for OPI in java because of its portability among different platforms. Sharing codes with other laboratories is one of the most important benefits of using Java/CORBA. Visual Cafe has been tested to create GUIs using beans component. We also plan to test more Java implementation of ORBs in both as a client and a server to augment the matrix for interoperation.

Introduction of CORBA 3 opens possibility to use MinimumCorba and RealTimeCorba for embedded systems. Interoperable naming service is now standard and could be used for object referencing in heterogeneous systems.

Object oriented database management system (OODBMS) may replace relational database management system (RDBMS) for CORBA based accelerator controls. Objectivity/DB for linux [10] is being tested for future evaluation.

5 CONCLUSIONS

It has been shown that replacing socket communication with a CORBA layer significantly improves manageability of accelerator controls.

There seems to be no significant problem associated with VisiBroker on VxWorks. For pSOS version, there seem to be more problems, but it may also be due to our lack of experience on this OS so far.

Interoperability between ORBacus and VisiBroker is found to be satisfactory and response times do not vary much with different combinations. Without using product dependent CORBA services, flexibility of the system codes can be improved at the expense of finding the way to share the stringified objects.

It is certain that CORBA is useful for integrating legacy controls for RIBF

REFERENCES

[1] Y. Yano and et al., Proc. of PAC 97, 930.
[2] http://www.borland.com/visibroker/
[3] http://www.windriver.com/products/html/vxworks.html
[4] http://www.isi.com/
[5] http://www.microsoft.com/
[6] http://www.ooc.com/
[7] http://www.omg.org/
[8] http://www.avaldata.co.jp/jpn/index.html
[9] http://elc.aska.or.jp/electro/index.html
[10] http://www.objectivity.com/