Performance Evaluation of Object Oriented Middleware

László Böszörményi[♣], Andreas Wickner[♠], and Harald Wolf[♣]

[♣]Institute of Information Technology, University of Klagenfurt
Universitätsstr. 65 – 67, A – 9020 Klagenfurt and
[♠]software design & management GmbH & Co. KG
Thomas-Dehler-Straße 27, D – 81737 München
email: laszlo@itec.uni-klu.ac.at, andreas.wickner@sdm.de, hwolf@edu.uni-klu.ac.at

A method for evaluating several aspects of the performance of object oriented middleware is introduced. Latency, data transfer, parameter marshalling and scalability are considered. A portable benchmark toolkit has been developed to implement the method. A number of actual middleware products have been measured, such as C++ and Java based CORBA implementations, DCOM and Java/RMI. The measurements are evaluated and related to each other.

1 Introduction

Currently, a certain number of de facto and de jure standards of object oriented middleware exist; also a number of implementations are available. Therefore, performance benchmarking is helpful in order to support the decision process at the beginning of the software lifecycle.

Existing measurements are rare and often restricted to a few aspects. Most of them ignore scalability either entirely or they consider the case of many objects on the server side only, ignoring the case of many clients attacking the same server concurrently (see [SG96], [PTB98], [HYI98]). The following research of the performance of a number of actual implementations is based on the most relevant object oriented middleware proposals, such as CORBA [OMG98], DCOM [Mic98] and Java/RMI [Sun98]. The evaluation covers several important performance aspects: latency, data transmission time and scalability.

A key issue in benchmarking are comparisons. Therefore, we have developed a portable benchmark toolkit.

2 Measurement Criteria and Methods

If we want to find out how fast our middleware is, we first have to inspect the latency and the turnaround time in detail (i.e. parameter marshalling/unmarshalling). A simple black box measurement of the turnaround time of parameterless remote method calls is sufficient to inspect the latency. White box measurements were performed with the help of interceptors [OMG98] on remote method calls with parameters of different type and size to measure data transmission time in detail (with measurement points before and after marshalling resp. unmarshalling).

Unfortunately, the standardization of interceptors in CORBA is still in progress. At this time, different manufacturers provide different interceptors (e.g. Orbix Filter, Visibroker interceptors) or do not support interceptors at all (e.g. TAO, JDK 1.2). Moreover, white box measurements are only meaningful if the resolution of the time

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 258-261, 1999.

measurement is in order of magnitude of microseconds and not milliseconds as usual in Java based systems.

Second, we want to know how the system scales. Special interest has been raised in the question: How does the server cope with a large number of concurrent client requests? It is not trivial to generate many parallel requests in order to evaluate scalability. Because it is hard to handle a large number of client processes (e.g. 1000 clients at once) we preferred to use threads.

3 The Benchmark Toolkit

Since CORBA is a standard and not a product, many implementations exist. The following relevant products have been investigated: Orbix 2.3 (C++) and OrbixWeb 3.0 (Java) from Iona, Visibroker 3.2 (C++ and Java) from Inprise (Borland/Visigenic), TAO 1.0 (C++) from D. Schmidt (Washington University) and JDK 1.2 (Java) from Sun (Beta3). The main focus was on CORBA implementations, because it is a widely used object oriented middleware platform. Moreover, CORBA provides effective support for a portable benchmark toolkit and for white box measurements. Additional measurements with Java/RMI and DCOM were made with separate benchmarks to compare at least basic functions (such as method invocation and object creation). All Java environments but JDK 1.2 included a JIT compiler.

A portable benchmark toolkit was developed by using the new Portable Object Adapter (POA) [SV97]. This enabled enhanced portability of source code and generated code frames (stubs, skeletons) between different CORBA implementations. Unfortunately, the POA is still not widely supported. A wrapper object was developed for the C++ based implementations, in order to hide the differences between different implementations.

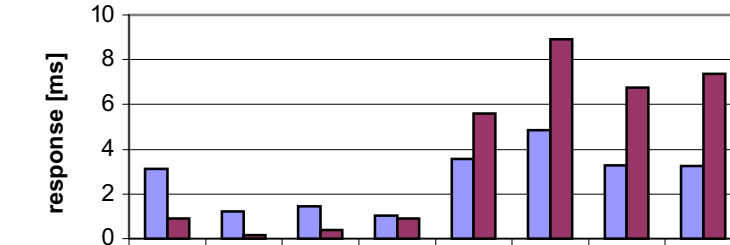
The benchmark information is specified and stored in a scenario profile, which contains a list of actions with parameters, such as: number of repetitions, amount of data, measuring offset, etc. The toolkit produces a workload on the system under test according to the profile. A transformation utility computes from the measured data the following indices: response time, marshalling overhead (client, server) and throughput. Minimum, maximum, mean and standard deviation are calculated for all indices.

For the measurements we used good, off-the shelf components. The server used in most cases was a NT host (Dual Pentium II, 233 MHz, 128 MB RAM, Windows NT 4.0, Service Pack-3,Visual C++ 5.0), and the client was a Solaris-Host (Sun Ultra 5,2, UltraSparc 300MHz processors, 256 MB RAM, Solaris 2.5, Sun Workshop C++ Compiler 3.0) connected by a 100 Mbit/sec Ethernet switch. For the DCOM measurements an NT system was used to host the clients as well.

4 Numeric Results

The following section shows a small excerpt from the evaluation. A summary of the entire investigation can be found in [BWW99]. The measurements illustrated in Fig. 1 show the response times for clients invoking remote methods without parameters and results. In each case, client and server are implemented using the same product. The C++ products are about 3 to 4 times faster than the Java based systems. The effect of the JIT compiler seems to be almost negligible in this measurement. The heavy fluctuation in the standard deviation in the Java implementations is caused by the

garbage collection. DCOM is fast, mainly because it is fully integrated into the Windows NT operating system.



	Orbix C++	TAO	Visibroker C++	DCOM	OrbixWeb	JDK 1.2	Visibroker Java	RMI
mean	3,119	1,212	1,444	1,02	3,555	4,844	3,264	3,237
standard deviation	0,896	0,167	0,401	0,892	5,603	8,898	6,745	7,377

Fig. 1. Method invocation without parameters

The following table shows the response times of transmitting 10 Kbyte of data within a sequence of shorts including the accumulated marshalling and unmarshalling subsections. TAO and DCOM do not support interceptors, thus this comparison contains only Visibroker and Orbix.

	Visibroker	Orbix	Sockets
Server	1017 µs	4058 µs	2975 µs
Net transfer	2402 µs	2378 µs	
Client	843 µs	7301 µs	

Visibroker is fast and has only little overhead compared with pure socket transmissions (43%). Orbix is astonishingly slow, its overhead hardly acceptable (362%). The reason is, unfortunately, unclear. Older Orbix versions were much faster than the current one (Iona has recently shipped Orbix 3.0 which is reported to include fixes for various performance bottlenecks but we have not yet been able to benchmark this version of Orbix.). Java-based products (not shown here) are about 3 to 4 times slower than the C++ based ones in transmitting small sequences, and more than 10 times slower in the case of large sequences. If sequences of complex data types, like nested structures are transmitted, the Java-based systems perform even worse.

Beside investigations on object creation (not shown here) some measurements have been made with special multithreaded CORBA clients (see Fig. 2). The client process is an Orbix-Client on the Solaris workstation that produces heavy load on the server process at the NT-Server. (For technical reasons, TAO, DCOM and Java/RMI were not considered in this test.) Depending on the number of clients the throughput grows to a maximum, then remains at the same level until it slowly decreases. The JDK 1.2 server stops working, when more than 197 client connections are open - that might be a bug in the Beta3 version. Orbix is quite slow, presumably due to heavy lock contention. The server of Visibroker for Java consumes so much memory, that Windows NT stops the server process at 800 open client connections.

5 Conclusions and Future Work

We demonstrated that it is possible to measure and compare several aspects of the performance of different middleware software. A portable benchmark toolkit was developed that can be used repeatedly for the same products and that can also be easily adapted to entirely new products. A number of useful actual measurements and comparisons were presented. The absolute numbers of some products, such as Visibroker, DCOM and TAO are quite promising by showing that the overhead caused by standard middleware is not necessarily high.

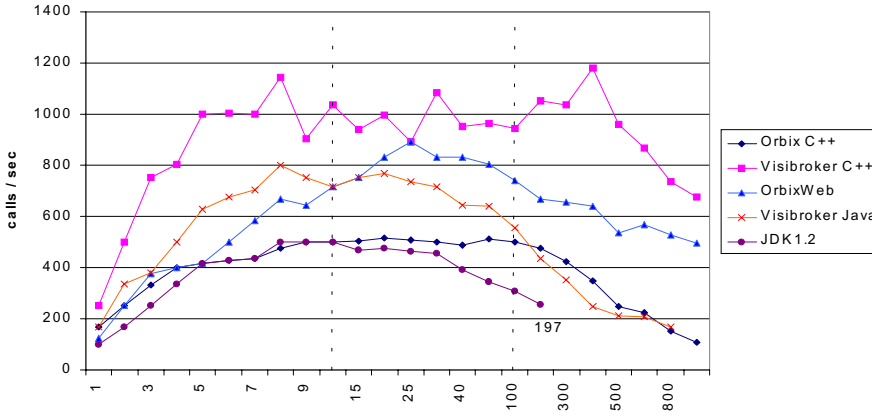


Fig. 2. Throughput with multiple Clients

In future, white box benchmarks could be extended to give more explanations of performance shortages. Emerging benchmark standards of OMG should be included.

6 References

[BWW99] L. Böszörményi, A. Wickner, H. Wolf: *Performance Evaluation of Object Oriented Middleware – Development of a Benchmarking Toolkit*, Technical Reports of the Inst. of Inf. Technology, University Klagenfurt, TR-9902, http://colossus.itec.uni-klu.ac.at/~laszlo/pubrec.html

[HYI98] S. Hirano, Y. Ysu, H. Igarashi: *Performance evaluation of Popular Distributed Object Technologies for Java*, ACM Workshop on Java for High-Performance Network Computing, February 1998

[Mic98] Microsoft Corporation: *DCOM Architecture*, White Paper, 1998

[OMG98] Object Management Group, Inc.: *The Common Object Request Broker: Architecture and Specification*, Revision 2.2, February 1998

[PTB98] Fantisek Plasil, Petr Tuma, Adam Buble: *CORBA Comparison Project*, Final Report, Distributed Research Group, Charles University, Prague, June 1998

[SV97] Douglas C. Schmidt, Steve Vinoski: *Object Adaptors: Concepts and Terminology*, SIGS C++ Report, Vol. 9, No 11., SIGS Publications, 1997

[SG96] Douglas C. Schmidt, Aniruddha Gokhale: *Measuring the Performance of Communication Middleware on High-Speed Networks*, Proceedings of the SIGCOMM Conference, August 1996

[Sun98] Sun Microsystems: *Java Remote Method Invocation Specification*, 1998