ARCHITECTURAL DESIGN AND PERFORMANCE ASPECTS OF DEVELOPING APPLICATIONS BASED ON MIDDLEWARE

Alexander Schill, Olaf Neumann, Christoph Pohl, Thomas Müller
TU Dresden, Fakultät Informatik, 01062 Dresden
{schill\|neumann\|pohl\|muellet} @ rn.inf.tu-dresden.de

Abstract For quite some time now, applications have been designed and developed in various projects of our research group by using middleware. In addition, various middleware products have been evaluated. An existing client/server system has been converted to EJB. The subject of this paper will be the results of these studies, along with the characteristics of the analyzed servers other concepts and related work.

Keywords: Enterprise JavaBeans, EJB, Performance

1. PERFORMANCE RESULTS

In the course of the performance analyses, the use of single servers and clustered application servers has been tested. Three commercial servers have been considered: Inprise Application Server [4], BEA WebLogic Server, and IBM Websphere.

As testing environment, the NT-Lab at the Chair of Computer Networks at the University of Science and Technology, Dresden, was used. This consists of several Dual-Pentium computers 766Mhz, 512MB that are interconnected via a switched 100Mbit LAN. The operating system used was Windows NT 4.0SP6a. On all application servers, an IBM Universal Database 7.0, powered by the jdbc.db2.net.Driver an XA-capable JDBC2.0 driver was used for efficient data access.

Special attention was given to providing as identical conditions as possible for testing the individual servers [1, 2, 3]. Therefore, the same test data and configurations were used.

Prior to a discussion of the performance comparison, some special aspects of the application servers shall be mentioned.

1.1. Deployment

All of the tested servers provided a graphical tool for creating deployment descriptors and supporting the actual deployment. In most cases, these tools are easy to use. However, they are not very efficient in the developing process of EJBs as the beans used in this process have to be re-deployed occasionally. In such cases it proved to be more helpful to perform script-controlled deployment, if that is supported by the server.

1.1.1 BEA Weblogic Server. To control the deployment, either a graphical tool or the command line can be used. The graphical tool should be used for the first configuration because it allows for an easy creation of the deployment descriptors. However, occasional crashs of this tool complicated routine work. Once the descriptors have been created, performing the deployment using the command-line can save a lot of time. To deploy the beans the next time the server is started, these have to be added to the file weblogic.properties. One could actually use the console for hot-deployment, or re-deploy already deployed beans at runtime, but these changes would only be valid while the server is running. Thus, it is always necessary to write the changes to the file weblogic.properties.

1.1.2 Inprise Application Server. Deployment is also performed with the help of the graphical console. The user can decide whether to comply more or less strongly with the J2EE standards. After deployment has been performed, it is possible to create a client-Jar containing the stubs needed. If a bean has been deployed, it exists even after the server has been restarted; it can only be removed if it is explicitly deleted from the server via the console.

1.1.3 IBM Websphere Application Server. Deployment in Websphere is rather slowly, both in regard to the steps necessary prior to deploying and the actual deployment process. The server allows deployment of several beans in one JAR simultaneously. In our tests, this simplification failed every time, so that each bean had to be deployed individually. After successful deployment, the beans are anchored in the server even after a restart. Websphere makes use of the serialized deployment-descriptors of EJB-version 1.0. As the descriptors have to be newly created, this interferes with one major objective of the J2EE standard: the reusability of the individual components. The internal deployment tool has been used to cope with this problem. This tool, however, is rather unstable and produces arbitrary errors. Moreover, changing the configuration requires a restart of the container, or the web-engine, which is rather time-consuming.

1.2. Clustering

All of the tested servers offer clustering to support fail-over and load-balancing. Even though these features have been differently realized on the respective servers, the basic concepts are very similar. Clustering can be performed on at least two levels. For one thing, it can be realized by distributing the requests of the servlets, i.e. on the web-level, for another, directly on the level of the EnterpriseBeans.

1.2.1 BEA Weblogic Server. Using replica-aware stubs, EJB's are automatically clustered on the Weblogic server. This can be easily configured. All one has to do is to start several servers so that they can be clustered. These servers must deploy the same beans under the same JNDI-names. It is important to remember that the ability to cluster is made possible by setting a property of the beans. For load-balancing, different procedures such as round-robin (standard), weight-based round robin, random, and parameter-based routing can be used. Parameter-based clustering requires the creation of a call-router that forwards the different calls to the corresponding server-instances.

1.2.2 Inprise Application Server. Clustering is realized via the JNDI-service. Only one instance of the name server is started in the cluster that is to be created. In order to achieve a fail-over of the name servers, these can also be started in a master-slave procedure. This distributes client requests among server instances according to the round-robin method. It is required that beans on the individual servers are addressable by identical JNDI-names and that the property vbroker.naming.propBindOn has been set to 1. Inprise supports clustering of Stateless Session Beans and claims to provide clustering of Statefull Session Beans via an additional Session State Service. However, the latter could not be verified in the current test, as no Stateful Session Beans were used.

1.2.3 IBM Websphere Application Server. The server supports clustering by means of an administrative database. All administration servers used in the cluster must use the same administrative database. Clones can be created based on a pre-configured application server. These clones can then be installed either on the same physical machine, or on other machines. In addition to the server clones, the EJBs must be WLM-enabled, i.e. so-called smart stubs for addressing the different servers must be installed on the client to allow load-balancing or fail-over, respectively. The methods round-robin and random can be used to distribute requests. These methods can be set to additionally include the option of preferring the local clone. Currently, IBM supports clustering of Stateless Session beans and servlets, or JSPs, while Statefull Session Beans are always assigned to a certain container.

1.3. Comparison

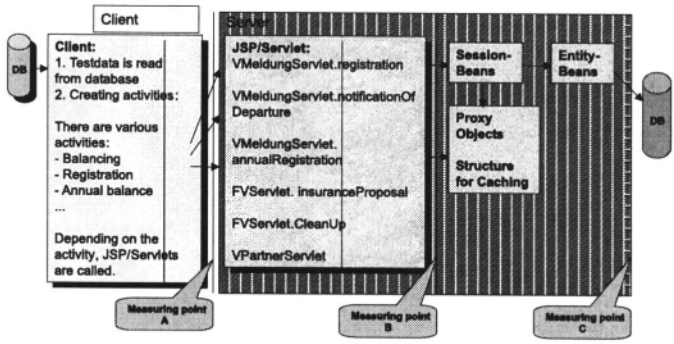

Figure 1. Measuring Points

For enabling a better evaluation of the various components' behavior, different measuring points have been implemented (see Figure 1).

Measuring point A determines the complete round trip time needed after an operation has been read from the database until the server has processed the request – including the time until to be displayed data has been received.

Measuring point B has been implemented in the servlets to determine processing times in the beans. This is actually not one single measuring point but rather several points where beans are called in the servlets.

Measuring point C determines the time needed for the internal processing in the database.

The figures of BEA and Inprise are about the same level. Inprise takes only slightly more time, and also shows leveling. Websphere consumes considerably more time. More precisely, IBM's round trip is longer especially for logic processing in the beans.

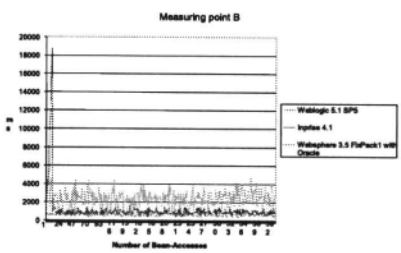

Figure 2. Results of Measure Point B.

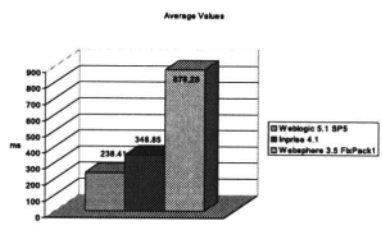

Figure 3. Average Values of Point C.

The diagram shown in Figure 3 summarizes the average measured results of the test in measuring point A. While Websphere needs about four times longer than BEA, Inprise's results are very similar to those of BEA. Even if the leveling behavior of Inprise is taken into account, it shows a slightly worse performance compared to BEA. In this example, measurings have been performed for 100 users. Unfortunately, even with such a small number server errors still occured. Inprise regularly stopped with an OutOfMemoryException that could not be corrected by increasing the memory assigned to the JVM, thus suggesting a server-internal memory problem. IBM could not complete some transactions because of internal, incomprehensible OSE problems. Both on IBM and BEA servers, tests have been run with 1,000 users. However, due to the errors results are not comparable.

2. RELATED WORK

This paper describes different aspects in the design process of applications with EJB. Furthermore, it gives details about the performance of a complex application. The comparison [11] and [12] as well as other benchmarks such as [10] give mostly an overview of server properties. Exact performance results can be found in [10]. Nevertheless, there are few publicly available facts about complex systems. A major problem about complex system evaluation is the mutual dependency of results, i.e. resources and components depend not only on equal component types. Although there are some good sources about patterns like [9], it is often not evident how to apply these concepts to real world applications based on EJB. A goal of this paper was to provide some contributions towards these aspects.

References

[1] Auswahl eines EJB-Applikations-Servers - Java-Spektrum 2/2000 S.12

[2] Special Interest Group - Enterprise JavaBeans - http://www.mgm-edv.de/ejbsig/

[3] Application Server Comparison Matrix - http://www.flashline.com/components/appservermatrix.jsp

[4] Inprise Applikation Server - Java Magazin 2/1999 S.71

[5] Read all about EJB 2.0 http://www.javaworld.com/javaworld/jw-06-2000/jw-0609-ejb.html

[6] Enterprise JavaBean Persistence 101 - http://www.sdmagazine.com/articles/2000/0004/0004b/0004b.htm

[7] Enterprise JavaBean Persistence 201 - http://www.sdmagazine.com/articles/2000/0004/0004c/0004c.htm

[8] Eberhard Wolff: EJB und das Java-Typsystem – Java Spektrum 6/2000 S. 62

[9] EJB DesignPatterns http://www.c2.com/cgi/wiki?EjbDesignPatterns

[10] http://nenya.ms.mff.cuni.cz/thegroup/EJBCOMP/ejb-public.pdf
[11] http://www.networkcomputing.com/1022/1022f2.html
[12] http://www.informationweek.com/759/java.htm