Load Balancing System using J2EE

Medi Suma
Student (M.Tech) , CSC,
Gokul Group Of Institutions
Visakhapatnam, India.

K.R.Koteeswa Rao
Asst. Prof, CSC,
Gokul Group Of Institutions
Visakhapatnam, India.

Abstract:

Distributed systems are the main architecture for enterprise applications. To develop a reliable distributed system, replication is necessary. The Adapt SIB repli cation system [27] provides a feasible solution for a reliable application server system. However, the system can not scale up since there is always only one primary server executing client requests. This thesis presents the LB system which is based on the Adapt SIB system, but with more functions.

The LB system may have the same number of server replicas as in the Adapt SIB system, but it can have more than one primary server, each being able to execute client requests. Thus, a load-balancing mechanism is needed to distribute the load equally among different replicas. In addi tion, reconfiguration in case of failure, and restart must be considered as well. This thesis presents load-balancing and reconfiguration solutions for the LB application server system.

Keywords:

J2EE, Replication System, Communication Systems, Database.

I. INTRODUCTION:

Traditional enterprise applications were designed as all-in-one modules. User interface, processing logic, and database access were tightly coupled. Such systems are hard to design, maintain and modify. With the rapid development of network ing technology, especially with the wide use of the Internet, the new generation of enterprise applications has a more feasible solution: multi-tier architecture.A multi-tier architecture [8] separates an application into several layers: client layer, business logic layer, and data layer. The client layer contains user interfaces, the business logic layer implements business rules on the retrieved data, and the data layer represents the underlying database. Each layer can be implemented as a self-contained component and deployed onto a separate machine.

Using a multi-tier architecture, each layer can be designed and maintained separately without affecting the functionality of the other layers. Another advantage of a multi-tier architecture is that the performance at each layer can be fine-tuned separately, hence providing better performance for the whole system. In other words, a multi-tier architecture makes distribution possible.

II. J2EE APPLICATION SERVER:

2.1 Introduction:

J2EE [22] stands for JAVA 2 Platform Enterprise Edition, which defines a stan dard for distributed component-based applications. It aims to provide a maintain able, reliable and scalable platform for enterprise applications. A J2EE application server is an example of a middleware server as we mentioned in Section 1.1. But let us first talk about Enterprise Java Beans (EJB) [23], which build the programmable units for J2EE application servers. Basically, there are two kinds of EJBs: session beans and entity beans. Session beans are used to implement business logic (for example, a program that implements a moneytransfer or that keeps track of all the goods a user has selectecd for purchase while she or he is logged into an online store).

There are two types of session beans: stateful and stateless beans. A stateful session bean is usually associated with a user session and keeps the information for this particular user during the period the user is connected to the system. A stateless session bean is used to perform arbitrary tasks, but will not keep state information once the task is finished.

2.2 Group Communication Systems:

Group communication systems [24], as implied by their name, provide communication for all members of an application group. The provide a number of messaging services to applications, and make reliable distributed systems possible.

III. THE ADAPT SIB REPLICATION SYSTEM

3.1 Introduction:

In this chapter, we talk about the Adapt SIB replication system [26, 27]. The Adapt SIB system serves as the base of the LB system, which will be discussed later. The main assumption in the Adapt SIB algorithm is that each client request generates exactly one transaction in the application server. That is, the execution of a client request r happens in the context of one individual transaction t. Hence there is a 1-1 association between transaction and request.

3.2 Correct Replication of J2EE Application Servers:

There are two things the replication algorithm has to guarantee in order to achieve fault-tolerance of a stateful J2EE application server. The first one is to guarantee the state consistency between the replicated application server and the backend database. State consistency means that if a transaction changes both the state of the application server and the database, the state of the application server and the state of the database are consistent after the transaction is committed (both have the state changes associated with the transaction) or aborted (none of the state changes remains at application server or database). Without replication and assum ing no failures, this state consistency is guaranteed by the transaction mechanism. But with replication, the state consistency among all the replicas of the application server must be guaranteed such that in case of failure, the backup replica can become the new primary without losing state consistency.

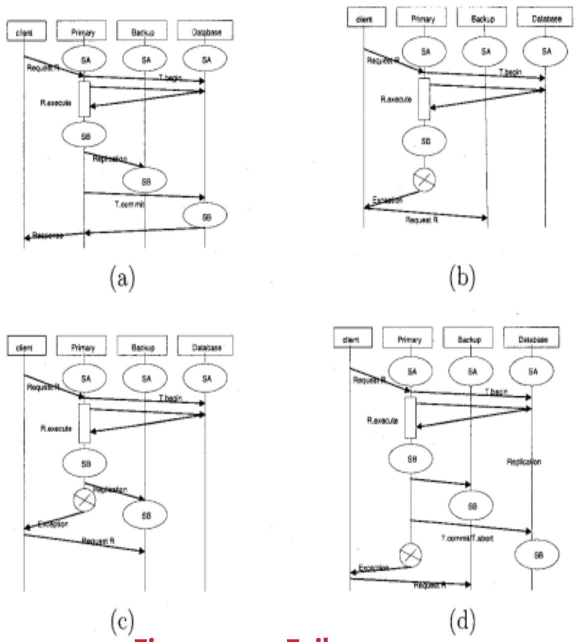


Figure 3 1: Failure cases

IV. LB SYSTEM DURING NORMAL PROCESSING

4.1 Design concept of the LB system:

The Adapt SIB system has one replication group with one primary and several backups. In the LB system, we have several such replication grOups, each with a primary and several backups. Each primary is able to handle client requests. Since there is more than one replication group in the system, we must have a load- balancing algorithm to dispatch a client to one of the replication groups, such that each replication group gets its share of the whole workload of the LB system.

But which server in the system will do the load-balancing work? A first solution is that we have a dedicated server, which works only as a load-balancer, also called dispatcher. This architecture is quite simple and easy to implement, but what will happen if the dispatcher fails? The whole system will be unavailable. Another solution is that we have a load-balancer group similar to the replication groups. Each LB server has a load-balancer which is a member of this group. But only one of them works as a primary load-balancer, all the others are only backup load-balancers. If the primary load-balancer fails, one of the backups takes over and becomes the new primary load balancer. Using a group of load-balancers has the advantage that if the primary fails, the GCS automatically detects the failure and can inform the backups.

V. RESULTS:

5.1 Introduction:

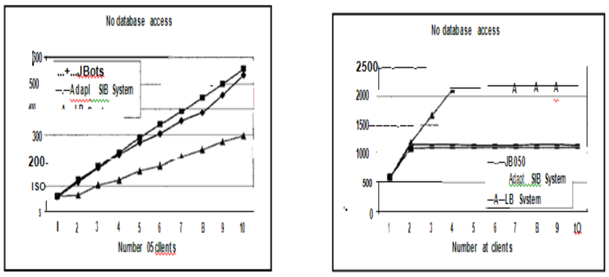
To evaluate the performance and the functionality of the LB system, we have conducted two sets of experiments. The first set of experiments compares the perfor mance of the original JBoss, the Adapt SIB system and the LB system during normal processing (without failure and recovery). The purpose of these tests is to see how load-balancing can improve the performance in case that the application server is the bottleneck of the system. Another set of experiments is to test the behavior of the LB system during failure and recovery. The failure cases are groupignore (a backup fails), groupupdate (the primary fails, and a backup takes over as primary), and grouprnerge (the primary fails and the group merges with another group). We would like to show the effect of each of these reconfigurations on the LB system.

The load-balancing algorithm used in all these tests is Round Robin.

5.2 Hardware and software used in the experiments:

Hardware:We used four Linux computers with the names cs8, cs9, cslO, csll (each has 3.4GHz Pentium 4 CPU with 1GB RAM). Three of them are used as LB servers, and one of them is used as a client simulator. They are all located in the same local network with a fast Ethernet connection. Software We compared three different configurations: the original Jfloss server, the Adapt SIB replication system, and the LB system. Furthermore, we had a client simulation program, which simulated the client access to the server.

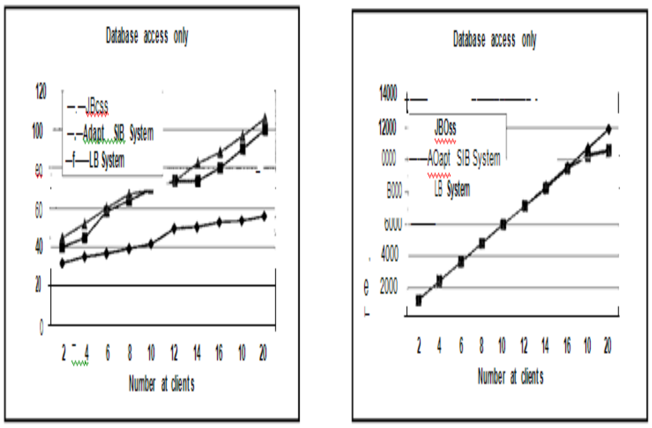
No database access



(a)Response time vs Number of clients (b)Throughput vs Number of clients

Figure 4—1: No database access

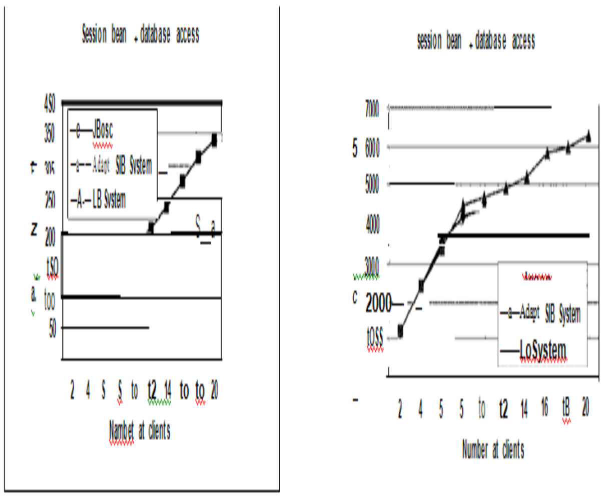
Database access only



(a)Response time vs Number of clients (b)Throughput vs Number of clients

Figure 4—2: Database access only

Database access plus SFSB processing



(a)Response time vs Number of clients (b)Throughput vs Number of clients

Figure 4—3: Database access plus SFSB processing

5.3 Performance tests during reconfiguration:

We show how the system behaves during reconfiguration (in cluding groupignore, groupnpdate, and groupmerge). groupignore means the failed server S only had backup RMs on it.

Groupignore:

Figure 6—4 shows the response time measured at the server side. At the begin ning, the system has only four clients, two for each replication group. csl1 fails at time slot 7 and then recovers at time slot 8. Because there were only backup RMs on server csll, the crash of csll does not affect any replication group. Hence, response times do not change during the reconfiguration.

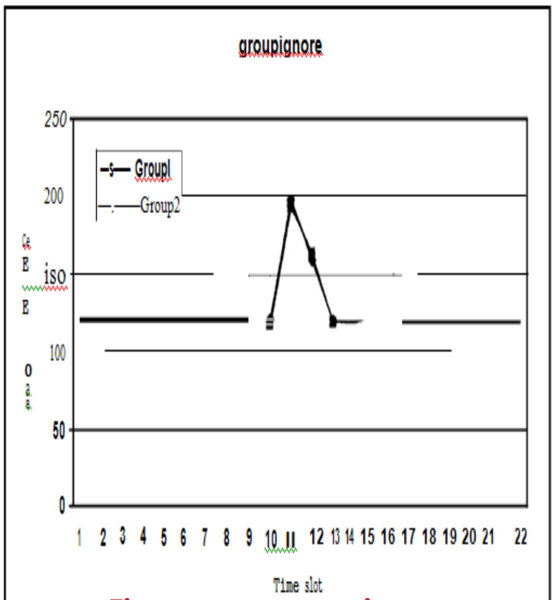


Figure 4—4: groupignore

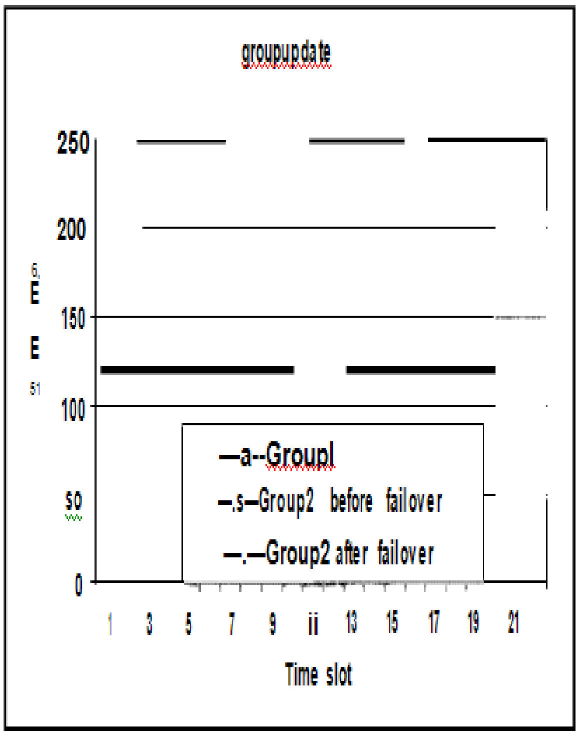


Figure 4—5: groupupdate

Figure 6—5 presents the response time measured at the server side when server cslO fails and recovers later. At the beginning, the system has only four clients, two for each group. At time slot 3, server cslO fails. Because the primary RM of group G2 was on server cslO, G2 has to find a new primary RM for its group. Since there are only backup RMs on server csll, and one of them is a backup RM of G2, the reconfiguration updates this backup RM to become the new primary RM of G2. After the reconfiguration, G2 continues to work as before except now the primary is on csll Gvoupmerge.

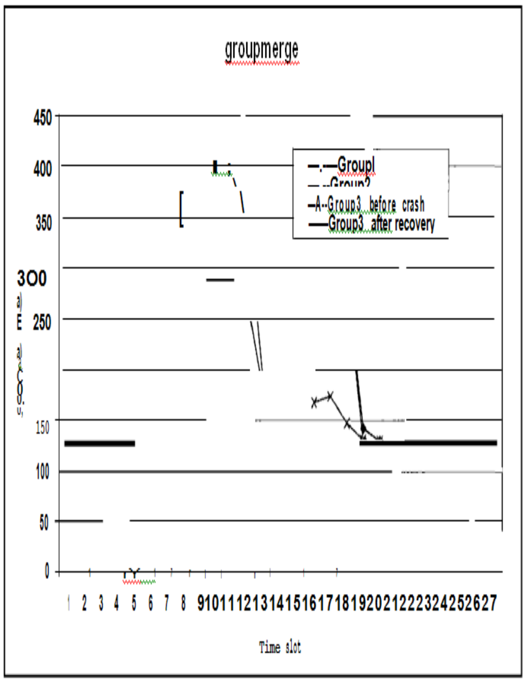


Figure 4-6: groupmerge

Figure 6—6 shows the response time during the reconfiguration measured at the server side. The first six clients start running in the system at time slot 1, each group has two clients, and they have almost the same response time. At time slot 4, server csll fails, which means group G3 has lost its primary RM. Since both cs9 and csll each has a primary RM, it is not possible in this case to do a groupupd ate reconfiguration, but a gronpmerge reconfiguration is performed.

CONCLUSION:

The current LB system provides load-balancing and performs reconfiguration automatically after failure and recovery. It is a feasible solution for the application server system.

REFERENCES:

1] Luis Aversa and Azer Bestavros. Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting. In 2000 IEEE International Performance, Computing and Communication Conference, 2000.

[2] J. Balasubramanian, D. C. Schmidt, L. Dowdy, and o. Othman. Evaluating the Performance of Middleware Load Balancing Strategies. In Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004.

[3] A. Bartoli, C. Calabrese, M. Prica, E. A. D. Muro, and A. Montresor. Adaptive Message Packing for Group Communication Systems. In OTM Workshops 2003:912-925, 2003.

[4] A. Bartoli, V. Maverick, S. Patarin, J. Vuãkovié, and H. Wu. A Framework for Prototyping J2EE Replication Algorithms. In mt. Symp. on Distributed Objects and Applications, 2004.

[5] BEA Systems Inc.BEA WebLogic Server Programming WebLogic Enterprise JavaB cans, Release 7.0 edition, September 2002.

[6] Birman, K. P., and R. Van Renesse. Reliable Distributed Computing with Isis Toolkit. IEEE, 1993.

[7] N. Budhiraja, K. Marzullo, F.B. Schneider, and S. Toueg. The Primary-Backup Approach. In Distributed Systems. Second edition. ACM Press, 1993.

[8] G. Coulouris, J. Dollimore, and T. Kindberg. Distributed Systems Concepts and Design. Addison Wesley, 2001.

[9] D. Dolev and D. Malki. The Transis Approach to High Availability Cluster Communication. Communications of the ACM, 39(4):64—70, 1996.

[10] Roy Friedman and Daniel Mosse. Load Balancing Schemes for High-Throughput Distributed Fault-Tolerant Servers. In 16th Symposium on Reliable Distributed Systems (SRDS'97), 1997.

[11] S. Frølund and R. Guerraoui. A Pragmatic Implementation of e-Transactions.n Proc. of Symp. on Reliable Distributed Systems (SRDS), 2000.

[12] S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-Based Scal able Network Services. In 16th ACM Symposium on Operating System Principle, 1999.

[13] Sacha Labourey and Bill Burke. JBoss Clustering. The JBoss Group, 2002.

[14] Spread Concepts LLC, Center for Networking, and Distributed System (CNDS). Spread Toolkit. http://www.spread.org.

[15] V. Maverick. Object Model for Pluggable J2EE Replication Strategies. Technical report, Universitá di Bologna, Bologna, Italy, June 2003.

[16] L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, R.K. Budhia, and C.A. Lingley Papadopoulos. Totem: A Fault-Tolerant Multicast Group Communication Sys tem. Communications of the ACM, 39(4):54—63, April 1996.

[17] R. Van Renesse, K.P. Birman, and S. Maffeis. Horus: A Flexible Group Com munication System. Communications of the ACM, 39(4):76—83, April 1996.

[18] Andreas Schaefer.JBoss: An In-Depth Look at the interceptor Stack, 2002. http://www.onjava.com/pub/a/onjava/2002/07/24/jboss statck.html.

[19] K. Shen, H. Tang, T. Yang, and L. Chu. Integrated Resource Management for Cluster-based Internet Services. In Symposium on Operating Systems Design and Implementation, 2002.

[20] Scott Stark and The JBoss Group. JBoss Administration and Development Third Edition (3.2.x Series). The JBoss Group, August 2003.

[21] Scott M Stark and The JBoss Group. JBoss Application Server, 2002.

[22] SUN Microsystems Inc. JAVA 2 Platform Enterprise Edition Specification, vl.3, October 2000.

[23] SUN Microsystems Inc. EJB 2.0 Specification, November 2003.

[24] R. Vitenberg, I. Keidar, G. V. Chockler, and D. Dolev. Group Communication Specification: A Comprehensive Study. ACM Computing Surveys, 33(4), 2001