Implementing O2PL Protocols in a Middleware Architecture for Database Replication

J.R. Juárez*, J.E. Armendáriz*, J.R. González de Mendívil*, J.R. Garitagoitia*, F.D. Muñoz-Escoí†
*Dpto. Matemática e Informática, Universidad Pública de Navarra, 31006 Pamplona, Spain
Email: {jr.juarez, enrique.armendariz, mendivil, joserra}@unavarra.es
†Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, 46022 Valencia, Spain
Email: fmunyoz@iti.upv.es

I. Introduction

Database replication consists of maintaining multiple copies of data on separate computers in order to improve the overall availability and performance of a system. 02PL [1] is one of the first concurrency algorithms designed to ensure the consistency of the replicated data following the ROWAA policy [2]. Although O2PL showed several advantages when compared with other general concurrency control approaches [1], it is known that unfortunately this protocol is prone to suffer distributed deadlocks, as it does not rely on any communication order guarantees.

In order to avoid this, we have implemented an O2PL based replication protocol called BRP in a middleware architecture [3]. Using a middleware architecture results in an greater portability to other DBMSs, despite having to add some collection and management tasks that reduce the performance of the resulting system. Additionally, this architecture stores locally useful metadata related to the database transactions making for a dynamic deadlock prevention scheme based on transaction priorities. These priorities may be based on different parameters, such as timestamps, number of restarts or writeset size.

Since no message ordering guarantees are considered, BRP needs to use 2PC [4] atomic commitment protocol. In this protocol to achieve a global commit of a transaction, all the participants must vote to commit it. This vote is normally sent after executing the transaction in each remote node, as BRP does. Nevertheless, a total order for the execution of the transactions is established by these priorities, so that it is not necessary to wait for the transaction completion prior to send the vote message from the remote site. For this reason, the vote can be sent after priority checking, avoiding the time spent executing the transaction in the database.

Taking account of this idea, we have implemented another replication protocol called ERP. The aim of this paper is to prove that the performance optimizations of the ERP protocol are significant. For this purpose, section III shows several tests that prove the overall performance of the ERP variant to be better than that achievable with the BRP one and, additionally, it is also able to reduce the abortion rate of the BRP.

Details of these protocols and their recovery algorithms as well as their correctness proofs can be found in [5], [6].

II. System configuration

For all the experiments, we used a cluster of 8 workstations (Fedora Core 1, Pentium IV 2.8GHz, 1GB main memory, 80GB IDE disk) connected by a full duplex Fast Ethernet network. PostgreSQL 7.4 was used as the underlying DBMS to keep a database composed of 30 tables each containing 1000 tuples. Finally, Spread 3.17.3 was in charge of the group communication, providing our protocols with reliable broadcast communication primitives. The results consist of executing transactions varying some parameters to check out the system behavior. Table I summarizes the parameters used in the different experiments.

TABLE I
Parameters of experiments

Experiments	WL1	WL2	WL3	Conflicts
Database Size	30 tables of 1000 tuples each			
Tuple Size	appr. 100 bytes			
Number of Servers	5	2-8	5	5
Number of Updates Operations	5	10	5-25	5
Number of Clients	1-20	2-8	5	5
Submission Rate in TPS	10-35	10	10	20
Hot Spot Size	0%	0%	0%	varying

III. Experimental results

A. Workload Analysis I: Protocol Evaluation

In this first experiment, transactions consist of 5 update operations. Workload was increased steadily from 10 to 35 tps and, for each workload, several tests were executed varying the number of clients from 1 to 20 in the whole system. Since BRP has to wait for the update execution before sending the vote in a commitment process and ERP has not, ERP behaves generally better than BRP (see Figure 1). Only once the CPU resources were saturated, both their behaviors seem to match.

B. Workload Analysis II: Varying Number of Servers

This second experiment tries to test the scalability of both protocols. As some experiences show [2], replication protocols do not cope very well with increasing system sizes. We have evaluated their performance varying the number of servers

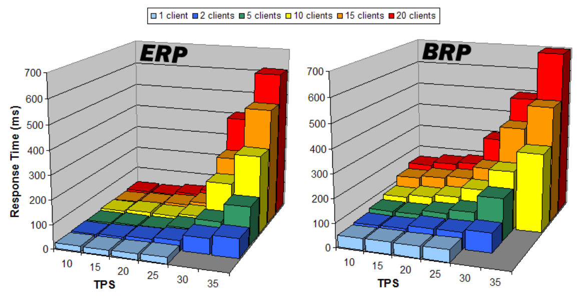

Fig. 1. Workload Analysis I: Protocol Evaluation

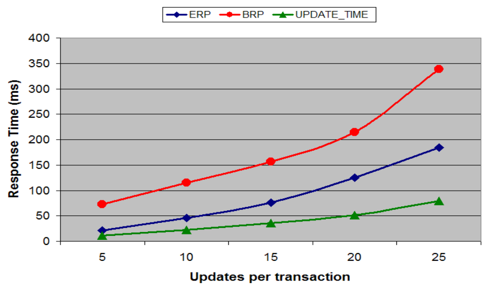

Fig. 3. Workload Analysis III: Varying Transaction Size

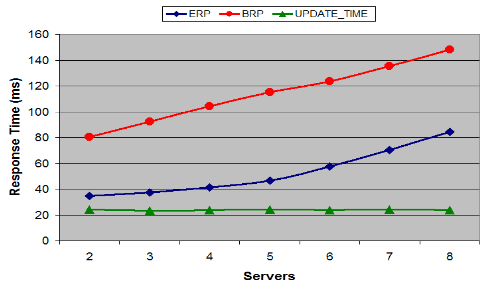

Fig. 2. Workload Analysis II: Varying Number of Servers

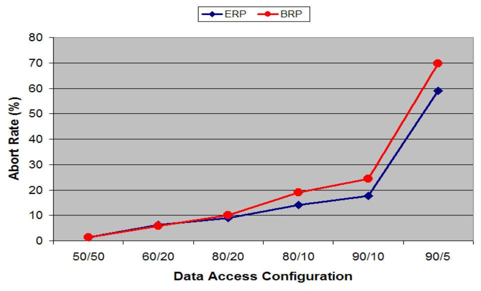

Fig. 4. Conflicts Analysis

from 2 to 8 and performing the same test suites in each configuration. A single client is allocated in each server and the load introduced into the system remains constant to 10 TPS in all the performed tests. As shown in Figure 2 response time increases as system size grows. Both protocols differ basically in the time spent before sending a vote when committing a remote transaction, which is mainly the update execution time. Since this time remains quite constant, both of them follow a similar behavior, at different levels, in terms of growing tendency.

C. Workload Analysis III: Varying Transaction Size

In order to have a better understanding of the difference between both protocols, in this test we tried to analyze this difference varying the number of operations included in a transaction, in other words, increasing transaction execution time. As update transaction time increases, the difference between both protocols becomes larger (see Figure 3), due to the fact that BRP has to wait for the worst transaction execution time of the remote machines and ERP only needs checking whether a conflict exists before sending a vote, without waiting for the transaction execution in the database.

D. Conflicts Analysis

In the previous experiments, conflict rates were rather small because we modeled a uniform data access distribution, so that the probability of conflict between two transactions was low. Therefore, considering that databases contain usually hot-spot areas that are accessed by most transactions, a hot-spot area of 1000 tuples was defined in the database and then we ran a sequence of tests varying the access distribution pattern to that area.

The access distribution is determined by the probability of operations accessing the hot-spot area (from 50% to 90%) and the percentage of tuples that will be accessed from the defined hot-spot area (from 50% to 5%). As it was expected, as probability of accessing the hot-spot area grew and its size decreased, conflict rates, and hence abort rates, became higher (see Figure 4).

Acknowledgment

This work has been supported by the Spanish Government under research grant TIC2003-09420-C02.

References

[1] Michael J. Carey and Miron Livny, “Conflict detection tradeoffs for replicated data.,” *ACM Trans. Database Syst.*, vol. 16, no. 4, pp. 703–746, 1991.
[2] Jim Gray, Pat Helland, Patrick E. O’Neil, and Dennis Shasha, “The dangers of replication and a solution.,” in *SIGMOD Conference*, H. V. Jagadish and Inderpal Singh Mumick, Eds. 1996, pp. 173–182, ACM Press.
[3] L. Irún-Briz, H. Decker, R. de Juan-Marín, F. Castro-Company, J.E. Armendáriz, and F.D. Muñoz-Escoí, “Madis: A slim middleware for database replication.,” in *Euro-Par*. 2005, Lecture Notes in Computer Science, Springer.
[4] Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman, *Concurrency Control and Recovery in Database Systems*, Addison Wesley, 1987.
[5] J. E. Armendáriz, F. D. Muñoz-Escoí, J. R. Garitagoitia, and J. R. González de Mendívil, “Design of a MidO2PL Database Replication Protocol in a Middleware Architecture,” Tech. Rep. ITI-ITE-05/09, Instituto Tecnológico de Informática, 2005.
[6] J. E. Armendáriz, J. R. Garitagoitia, J. R. González de Mendívil, and F. D. Muñoz-Escoí, “A Basic Replication and Recovery Protocol for the MADIS Middleware Architecture,” Tech. Rep. ITI-ITE-05/01, Instituto Tecnológico de Informática, 2005.