Recovery Protocols For Replicated Databases - A Minimal Survey [*]

Luis H. García-Muñoz, J. Enrique Armendáriz-Iñigo, Francisco D. Muñoz-Escoí
Instituto Tecnológico de Informática, Valencia, España
{lgarcia, armendariz, fmunyoz}@iti.upv.es
Technical Report ITI-ITE-06/07

In replicated databases, the same data are stored in different sites distributed in different places. In the last decade they have been widely used to increase performance, availability and fault tolerance. When a server site fails, the clients can redirect their transactions to another available one, but possibility of recovering sites that have failed must exist. When recovery protocols are designed it is necessary to consider the characteristics of the replication protocols developed.

In this work we make an analysis for the recovery protocols that were proposed in the last years, to determine the best protocol in each case and to identify the kind of problems they solve and the kind of problems that leave uncovered. With this information we intend, in a future work, the development of new recovery protocols that combine their best characteristics.

With the development of Group Communication Systems (GCS) and virtual synchrony [5], synchrony points between failures, sites recovery and the delivered set of messages to available sites can be generated. GCS provides membership service and reliable multicast. The membership service maintains a list of currently active and connected sites, providing with this the view concept.

The recovery task basically consists in transferring the information lost during the failure interval, from one or more active to one or more recovering sites, without altering the service capability of the system. The recovery protocol is usually integrated into the applied replication protocol and must consider:

1. The way database state is transferred: sending the entire database, using object versions or resending lost messages.
2. Concurrency control mechanism applied: optimistic or pessimistic, unique or distributed manager, and whether it is version based or not.
3. The way to distribute recovery work: it can be distributed or centralized.

[*]This work has been funded by the MCYT and the MEC under projects: TIC2003-09420-C02 and TIN2006-14738-C02.

According to [9] when the GCS is used to transfer the recovery information it is only possible to send the entire database, because the system does not know which data have been changed since failure occurs. It has the advantages of easy implementation and does not fully suspend system execution, only write operations are deferred. Its disadvantages are that database transfer is made under a transactional scheme, where a read lock is set on data until reception on recovering site is confirmed, and such lock is applied to the entire database. Additionally when a site was in a short-time failure full database transfer may be highly inefficient.

The second choice suggested in [9] is to use remote backup utilities from the database system. The following alternatives are proposed: a) Checking version numbers, b) Restricting the set of objects to check, c) Filtering the log, d) Lazy data Transfer. Additionally usage of Enriched View Synchrony (EVS) is proposed. Using EVS the reconfiguration process is encapsulated, and the database system has a more realistic picture of what is happening in the system.

In [6], recovery protocols based on GCS are proposed, when replication protocols: Broadcast Writes, Delayed Broadcast and Single Broadcast Protocol, described in [1] are used. These recovery protocols are: Single Broadcast Recovery, Broadcast Writes Recovery, and Delayed Broadcast Recovery, where two alternatives are presented: Log Update Method and Augmented Broadcast Method. The main advantage is that it can support the majority of distributed database transactions types without data versioning. The disadvantages for the first and third case is that additional work is given to on process transaction sites (avoiding the work on loggers) and that they require a change in their lock management algorithm. When the Log Update Method is used with Broadcast Writes, the Logger must remove messages from the log for transactions that are aborted for any reason, not only for those that have an explicit abort message or whose commit message was rejected.

In [8], parallel recovery is proposed by means of procedures that define conflict classes. Recovery is based on DB-partitions. Its advantages are that can be extended to work in parallel and with this, optimize the transfer time and distribute the recovering task overload. Transactions are not processed only in the view change, when sites are blocked. Its disadvantages are that transactions can only be executed on the partition master site. When the failure period is large, the amount of information to transfer will be abundant.

In [7], a lazy recovery protocol for a configurable replication protocol that can make updates eager, lazy or in a hybrid way is presented. The advantages of this protocol are: the recovery task is fully supported by the hybrid replication protocol, thus the recovery is part of the basic algorithm, updates are performed until the recovering site accesses stale data, there is no need of locking objects because it uses object versions and the transaction abortion rate is small. The disadvantage is: the transaction service time is usually greater than in pure lazy replication protocols.

The work proposed in [3] is a framework for reliable broadcast protocols that are used as a basis for database replication protocols. Its objective is to manage the logging of missed update messages in the broadcast protocol core, providing with this, automatical recovery in short-term failures, but discarding the log and notifying in case of long-term failures. The main advantages of this protocol are that it combines the version-based and log-based recovery depending on which one is the best to use, it does not restrict to one transference model and can take advantage of each one in its case. It is gained a minimal blocking time to replicas involved in log-based recovery. The disadvantages are that it is necessary to maintain related information to both recovery methods. Transaction service time is increased because all messages are stored in persistent storage.

FOBr [4] complements the replication protocol FOB, which is an optimistic protocol with eager updates that use membership service provided by GCS. Its advantages are that it minimizes the transferred information amount, its recovery work is distributed, it allows transaction execution during recovery period, and that it uses a small amount of space for storage. The disadvantages are that it complements a replication protocol with non standardized isolation levels; for each committed transaction, the objects in the WriteSet must be recorded if there was a failure site and this may reduce the performance during failure intervals.

J.E. Armendáriz in [2] proposes a recovery protocol for three eager update replication protocols. The general idea for recovery is based on forming a database dynamic partition (DB-partition) that contains the lost elements, grouped by lost views. The main advantages are that the recovery work is distributed, the DB-partition in a recoverer site can be released even when the recovery process is not concluded, transactions can be committed in recoverer sites while it has not interference on the DB-partition, recoverer site can start to accept transactions as soon as DB-partitions are set in it. The disadvantage is that if DB-partitions are defined on the basis of each view modified objects, an object may be transferred several times, to avoid this we must ”compact” the DB-partitioning.

Once this set of protocols has been surveyed, as a concluding remark we advise to consider recovery algorithms that use version-based management and that distribute the recovery work among available sites to balance the workload during the recovery process. Very few replicated database recovery systems are capable to combine these technics to reduce recovery times. When it has been partially possible (as in [8], [7]), it was because replication protocols had some special characteristic (the use of a primary copy schema in [8], that reduces flexibility and compromises fault tolerance; the use of lazy updates in [7], that may compromise consistency). The work presented in [2] could be a good exception, but it has not presented performance measurements that confirm its good theoretical performance. This analysis will be used as a basis for new recovery protocols design that try to combine the analyzed protocols advantages.

References

[1] D. Agrawal, G. Alonso, A. E. Abbadi, and I. Stanoi. Exploiting atomic broadcast in replicated databases. In *EuroPar*, pages 496–503. LNCS vol. 1300. Springer, 1997.

[2] J. E. Armendáriz. *Design and Implementation of Database Replication Protocols in the MADIS Architecture*. PhD thesis, Univ. Pública de Navarra, Pamplona, Spain, Feb. 2006.

[3] F. Castro, J. Esparza, M. I. Ruiz, L. Irún, H. Decker, and F. D. Muñoz. Clob: Communication support for efficient replicated database recovery. In *PDP*, pages 314–321, 2005.

[4] F. Castro, L. Irún, F. García, and F. D. Muñoz. Fobr: A version-based recovery protocol for replicated databases. In *PDP*, pages 306–313, 2005.

[5] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. In *ACM Computing Surveys 33(4)*, pages 1–43, 2001.

[6] J. Holliday. Replicated database recovery using multicast communication. In *NCA*. IEEE-CS Press, 2001.

[7] L. Irún, F. Castro, F. García, A. Calero, and F. Muñoz. Lazy recovery in a hybrid database replication protocol. In *XII Jornadas de Concurrencia y Sistemas Distribuidos*, 2004.

[8] R. Jiménez, M. Patiño, and G. Alonso. An algorithm for non-intrusive, parallel recovery of replicated data and its correctness. In *SRDS*, pages 150–159. IEEE-CS Press, 2002.

[9] B. Kemme, A. Bartoli, and Ö. Babaoglu. Online reconfiguration in replicated databases based on group communication. In *DSN*, pages 117–130. IEEE-CS Press, 2001.