# Database Replication Approaches

F. D. Muñoz, H. Decker, J. E. Armendáriz, J. R. González de Mendívil

Instituto Tecnológico de Informática

{fmunyoz,hendrik}@iti.upv.es, {enrique.armendariz,mendivil}@unavarra.es

Technical Report TR-ITI-ITE-07/19

# Database Replication Approaches

F. D. Muñoz, H. Decker, J. E. Armendáriz, J. R. González de Mendívil

Instituto Tecnológico de Informática

Technical Report TR-ITI-ITE-07/19

e-mail: {fmunyoz,hendrik}@iti.upv.es, {enrique.armendariz,mendivil}@unavarra.es

October 5, 2007

## 1 Introduction

Databases are fully replicated in order to get two complementary features: performance improvement and high availability. Performance can be improved when a database is replicated since each replica can serve read-only accesses without requiring any coordination with the rest of replicas. Thus, when most of the application accesses to the data are read-only, they can be served locally and without preventing accesses in the same or other replicas. Moreover, with a careful management, the failure of one or more replicas does not compromise the availability of the database.

Initially, database replication management was decomposed into two tasks: concurrency control and replica control, usually solved by different protocols. The solutions of the non-replicated domain were evolved into distributed concurrency control protocols [5], taking as their base either the two-phase-locking (2PL) or some timestamp-ordering protocol. Replica control management was based on voting techniques [11]. These voting techniques assign a given number of votes to each replica, usually one, and require that each read access collects a read quorum (i.e., "r" votes) and each write access a write quorum (i.e., "w" votes). The database must assign version numbers to the items being replicated, and the values of "r" and "w" must ensure that r+w is greater than the total number of votes, and that "w" is greater than a half of the amount of votes. Thus, it can be guaranteed that each access to the data reaches at least one copy with the latest version number for each item. This ensured consistency, but the communication costs introduced by these techniques were high.

Voting replica-control protocols were still used in the next decade, boosting their features and including also management for system partition handling when dynamic voting approaches were included [14].

However, replication management is not so easy to achieve when both concurrency and replica control are merged, since what the replica control protocols do for ensuring consistency has to be accepted by the concurrency control being used. Deadlocks and transaction abortions are common when both protocols are joined. So, it seems adequate to find better solutions for this global management, i.e., replication protocols that consider both concurrency and replica controls. A first example of this combined technique is [21], where a special kind of voting algorithm is combined with timestamp-based concurrency control. However, his solution still relies on simple communication primitives, and is not efficient enough, both in terms of response time and abortion rate. Note that efficient total order broadcast protocols were not produced until the middle eighties [6], and they could not be used in these first stages.

So, new replication techniques were introduced for databases, as an evolution of the process replication approaches found in distributed systems. Thus, depending on the criteria being used, several classifications are possible [12, 22]. The proposal of [23], distinguishing five different techniques (active, weak-voting, certification-based, primary copy and lazy replication) will be followed here. In all these techniques, the protocols need a reliable total order broadcast in order to propagate the transaction updates. This is the regular way for achieving replica consistency in any distributed application.

## 2 Active Replication

In the active replication technique, the client initially submits a transaction request to one of the replicas. Such delegate replica broadcasts the request to all replicas and all of them process the transaction from its start. Note that different transactions can use different delegate replicas. This technique requires complete determinism in the execution of a transaction, since otherwise the resulting state could be different among replicas. In order to easily develop this model, transactions can be implemented as stored procedures. Note that all transaction operations or parameters should be known before such transaction is started, being propagated in its starting broadcast.

The main advantage of this technique is that the support for executing transactions can be the same in both delegate and non-delegate replicas. On the other hand, its disadvantages consist of (1) requiring determinism in the transaction code, and (2) compelling read operations to be executed in all replicas, losing the possibility of balancing reads among replicas, and thus compromising one of the best performance improvements of database replication.

This technique is a direct translation of the active replication model for distributed systems. It has not been directly used for database replication, but there are some adaptations that have eliminated several of its intrinsic problems, improving its performance and behavior. One of such approaches is the Database State Machine [20] that uses deferred updates, i.e., it delays the synchronization/broadcast point until commit is requested in the delegate replica.

## 3 Weak-Voting Replication

In the weak-voting replication technique, the delegate replica initially executes the complete transaction, and when the application requests its commitment, its writeset is collected and broadcast to all replicas (including the delegate one). Once such writeset is delivered, the replication protocol evaluates if there is any conflict with any previously committed transaction. If so, the transaction is aborted. Otherwise it is accepted and committed. Once the termination decision has been taken, a second message is broadcast communicating to the other replicas its result (commit or abort). The non-delegate replicas can not decide by themselves, and they should wait such a second broadcast in order to complete such transaction.

The advantages of this technique are: (1) no readset should be broadcast in order to decide the outcome of a transaction, since read-write conflicts can be evaluated in the delegate replicas; i.e., the readset of the local transaction against the writesets of the remote ones, (2) read-only transactions can be locally executed and completed in their delegate replicas, without needing any broadcast. On the other hand, its main inconvenience consists in needing two broadcasts in order to complete an updating transaction, although the second one does not need to be totally ordered. Note that the other techniques only need a single broadcast per transaction.

This technique is particularly suitable for replication protocols ensuring the serializable isolation level, since such level needs to evaluate read-write conflicts, and they do not demand readset propagation in this technique. The SER protocol of [15] is a good sample of this kind of replication approach.

## 4 Certification-Based Replication

The certification-based technique shares some characteristics with the weak-voting one, but using a symmetrical transaction evaluation approach; i.e., letting each replica to individually decide the outcome of each transaction. To this end, this technique needs to total-order broadcast the transaction readset and writeset when its commit is being requested by the user application. As a result, when such message is delivered, all replicas share the same historic list of delivered messages and can look for conflicts with concurrent transactions in the same way. This eliminates the need of a second reliable broadcast for announcing the outcome of each transaction. Note, however, that this validation/certification process needs both readsets and writesets in order to work; at least, when read-write conflicts should be evaluated.

Hopefully, not all isolation levels demand read-write conflict evaluation. Indeed, the *snapshot isolation* [3] level (or SI, for short) only needs to check for write-write conflicts. So, the certification-based technique

has been mainly used in order to provide such isolation level. Examples of this kind of protocols are [9, 16, 18].

# 5 Primary Copy Replication

In the primary copy replication technique all transactions are forwarded to and executed by a single replica, the primary one. The other replicas are only its backups or secondaries, and apply the updates of writing transactions before they are committed in the primary. Since this approach easily overloads the primary replica, read-only transactions can also be applied in secondary replicas, thus balancing the load.

At a glance, the main problem of this technique is its lack of scalability, since updating transactions should be executed by a single replica and this compromises its performance. However, this also introduces some performance gains, since this kind of effort removes the need of coordination among multiple replicas in order to decide the outcome of each transaction. Moreover, since local concurrency control can be used, conflicting transactions improve their probability of success since a pessimistic concurrency control (e.g., 2PL) can be employed.

Indeed, [19] have used the primary copy technique in order to increase the performance of a middleware-based data replication system. To this end, they divide the database into a set of conflict classes, and assign a master replica to each of such classes; i.e., playing the primary role for such conflict class. Each incoming transaction is forwarded by its delegate replica to its associated master replica, once the items to be accessed are known. Thus, the load can be easily balanced, and the concurrency control can be locally managed by each master replica. As a result, the performance is highly increased.

# 6 Lazy Replication

Lazy replication propagates the updates of a transaction once it has already committed. This allows a fast transaction completion, but does not always ensure replica consistency and may lead to a high abort rate. Despite its disadvantages, this technique has been used in several commercial DBMSs and it is the main option when mobile or disconnected databases are considered.

If any replica can update directly its local data, transmitting later the updates to other replicas, concurrent load may lead to a high abort rate. For concurrency control purposes, a timestamp-based solution can be used. There have not been many replication protocols of this kind, being [13] one of the exceptions. However, its solution is not completely lazy, but hybrid, since the number of replicas that receive the updates before commit time is configurable. In its purely lazy configuration, the abort rate is reduced using an empirical expression that forecasts the probability that a data item was outdated, before it is locally accessed. All computations needed in these expressions use locally collected data. If such a probability exceeds a dynamically-set threshold, the data item is updated from its owner replica. Such owner replica always maintains the latest version of the item. Different items may have different owner replicas. Note that when the protocol arrives to its voting termination stage, the probability of success is improved.

In lazy primary copy protocols, since the accesses are always managed by the same replica, such a replica may use any local concurrency control approach for avoiding conflicts between transactions. Primary copy solutions have been used in some commercial database systems, for instance Sybase Replication Server 11 [7]. In these systems two trends can be found. The first one uses replication to ensure only availability, not to improve performance. In such cases, the replicas behave as standby copies of the primary replica. In the second one, replication is mainly used to enhance performance, and serializable consistency is not maintained. Note that most applications can be perfectly run with relaxed consistency modes, or isolation levels. Indeed, the default isolation level of most relational DBMSs is not “serializable”, but “read committed” (for instance, in PostgreSQL). Another sample of lazy primary copy replication protocol is the one being described by [8] for providing the snapshot isolation level.

# 7 Future Trends

Several emerging lines of research in the database replication field can be distinguished:

**Load balancing** : In order to improve performance, one of the key issues consists in balancing the load being received by each server replica. Several approaches have been tried in this research line, considering different parameters that rule the replica assignment: memory usage [10], transaction lengths [17], transaction scheduling [1], etc. One of the promising approaches in load balancing is the distribution of transactions based on their conflicts. Thus, inter-conflicting transactions are placed in the same node, reducing their probability of abortion, since they will be managed by a local (and pessimistic) concurrency control mechanism. Additionally, non-conflicting transactions can be placed on lightly loaded replicas. The combination of both approaches has generated good results in [24].

**Support for multiple isolation levels** : Current database replication protocols have focused on supporting a single isolation level, but DBMSs have always supported multiple levels. This allows the application designer to select the most appropriate level for each transaction, improving their response time, since transactions that use relaxed isolation levels seldom get blocked. Our groups have published some initial works in this area [4, 2].

**Scalability** : Although there have been some interesting results [19] improving the scalability of data replication systems, further work is still needed in this area.

# 8 Conclusion

Database replication had commonly used lazy protocols in commercial DBMSs, ensuring thus good performance, but without guaranteeing full replica consistency. This may lead to some transaction losses in case of failure. To overcome these problems, eager replication protocols with group communication support have been proposed in the last decade. The use of total order broadcast protocols has allowed the development of new kinds of eager replication: weak-voting and certification-based techniques. Such solutions are able to ensure one-copy serializability with a performance similar to lazy protocols, with better (although still limited) scalability and lower abort rates. However, these solutions are not applied to commercial database management systems, yet.

Future trends in database replication research should focus on providing support for multiple isolation levels at once, and on further improving system scalability.

# References


[1] C. Amza, A.L. Cox, and W. Zwaenepoel. A comparative evaluation of transparent scaling techniques for dynamic content servers. In *International Conference on Data Engineering*, pages 230–241, 2005.

[2] J.E. Armendáriz-Íñigo, J.R. Juárez, J.R. González de Mendívil, H. Decker, and F.D. Muñoz-Escoí. k-bound gsi: A flexible database replication protocol. In *ACM Annual Symposium on Applied Computing*, pages 556–560, 2007.

[3] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A critique of ANSI SQL isolation levels. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 1–10, San José, CA, USA, May 1995.

[4] J.M. Bernabé-Gisbert, R.Salinas-Monteagudo, L. Irún-Briz, and F.D. Muñoz-Escoí. Managing multiple isolation levels in middleware database replication protocols. *Lecture Notes in Computer Science*, 4330:710–723, 2006.

[5] P.A. Bernstein and N. Goodman. Concurrency control for distributed database systems. *ACM Computing Surveys*, 13(2):185–221, 1981.

[6] K.P. Birman and T.A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):47–76, 1987.

[7] R. Breton. Replication strategies for high availability and disaster recovery. *IEEE Bulletin of the Technical Committee on Data Engineering*, 21(4):38–43, 1998.

[8] Khuzaima Daudjee and Kenneth Salem. Lazy database replication with snapshot isolation. In *International Conference on Very Large Data Bases*, pages 715–726, Seoul, Korea, September 2006. ACM.

[9] S. Elnikety, W. Zwaenepoel, and F. Pedone. Database replication using generalized snapshot isolation. In *SRDS*, pages 73–84, Orlando, FL, USA, October 2005.

[10] Sameh Elnikety, Steven G. Dropsho, and Willy Zwaenepoel. Tashkent+: memory-aware load balancing and update filtering in replicated databases. In *EuroSys*, pages 399–412, 2007.

[11] D. K. Gifford. Weighted voting for replicated data. In *Proc. of the 17th ACM SIGOPS Symposium on Operating Systems Principles*, pages 150–159, Pacific Grove, CA, USA, December 1979. ACM.

[12] Jim Gray, Pat Helland, Patrick E. O'Neil, and Dennis Shasha. The dangers of replication and a solution. In *SIGMOD Conference*, pages 173–182, 1996.

[13] Luis Irún-Briz, Francesc D. Muñoz-Escoí, and José M. Bernabéu-Aubán. An improved optimistic and fault-tolerant replication protocol. In *DNIS*, pages 188–200, 2003.

[14] S. Jajodia and D. Mutchler. Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Trans. on Database Sys.*, 15(2):230–280, June 1990.

[15] B. Kemme and G. Alonso. A new approach to developing and implementing eager database replication protocols. *ACM Transactions on Database Systems*, 25(3):333–379, September 2000.

[16] Y. Lin, B. Kemme, M. Patiño-Martínez, and R. Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *SIGMOD Conference*, pages 419–430, 2005.

[17] Jesús M. Milán-Franco, Ricardo Jiménez-Peris, Marta Patiño-Martínez, and Bettina Kemme. Adaptive middleware for data replication. In *Middleware*, pages 175–194, 2004.

[18] Francesc D. Muñoz-Escoí, Jerónimo Pla-Civera, María Idoia Ruiz-Fuertes, Luis Irún-Briz, Hendrik Decker, José Enrique Armendáriz-Iñigo, and José Ramón González de Mendívil. Managing transaction conflicts in middleware-based database replication architectures. In *SRDS*, pages 401–410, 2006.

[19] M. Patiño-Martínez, R. Jiménez-Peris, B. Kemme, and G. Alonso. MIDDLE-R: Consistent database replication at the middleware level. *ACM Trans. Comput. Syst.*, 23(4):375–423, 2005.

[20] F. Pedone, R. Guerraoui, and A. Schiper. The database state machine approach. *Distributed and Parallel Databases*, 14(1):71–98, 2003.

[21] R.H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, 1979.

[22] M. Wiesmann, A. Schiper, F. Pedone, B. Kemme, and G. Alonso. Database replication techniques: A three parameter classification. In *Proc. of the 19th IEEE Symposium on Reliable Distributed Systems*, pages 206–217, October 2000.

[23] Matthias Wiesmann and André Schiper. Comparison of database replication techniques based on total order broadcast. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):551–566, April 2005.

[24] V. Zuikeviciute and F. Pedone. Conflict-aware load-balancing techniques for database replication. Technical report, Univ. of Lugano, Lugano, Switzerland, 2007.