Database Replication Protocols

Francesc D. Muñoz-Escoi
Instituto Tecnológico de Informática, Spain

Luis Irún-Briz
Instituto Tecnológico de Informática, Spain

Hendrik Decker
Instituto Tecnológico de Informática, Spain

INTRODUCTION

Databases are replicated in order to obtain 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 without preventing other processes to access the same or other replicas. Moreover, a careful coordination management can ensure that the failure of one or more replicas does not compromise the availability of the database as long as at least one of the replicas is alive.

BACKGROUND

Initially, database replication management had been decomposed into two tasks: concurrency control and replica control, usually tackled by different protocols. In the first case, solutions known from the non-replicated domain have been evolved into distributed concurrency control protocols (Bernstein & Goodman, 1981), based either on the two-phase-locking (2PL) or some timestamp-ordering protocol. In the second case, replica control management was based on voting techniques (Gifford, 1979). These voting techniques assign a given number of votes to each replica, usually one, requiring that each read access collects a read quorum (“r” votes) and each write access a write quorum (“w” votes). The database must assign version numbers to the records being replicated so that sum of the values of “r” and “w” is ensured to be greater than the total number of votes and that “w” is greater than half the amount of votes. Thus, it can be guaranteed that each access to the database reaches at least one copy of the latest version number for each record. This approach ensured consistency, but the resulting communication costs could be prohibitively high.

Voting replica-control protocols with improved features were still used in the next decade, also including routines for system partition management using dynamic voting approaches (Jajodia & Mutchler, 1990).

However, replication management is not easily achieved when concurrency and replica control are merged, since what the replica control protocols do for ensuring consistency has to be accepted by the used concurrency control. Unfortunately, deadlocks and transaction abortions are quite common when both protocols are merged. Thus, it seems adequate to find better solutions for this global management task, that is, replication protocols that cater for both concurrency and replica controls. A first example of this combined technique is Thomas (1979), 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. In fact, efficient reliable or atomic broadcast protocols were not produced until the mid-1980s (Birman & Joseph, 1987) and could not yet be used in these first stages.

Thus, new replication techniques were introduced in the database arena, having evolved from the process replication approaches known from distributed systems. Depending on the used criteria, several classifications are possible. However, it is useful to distinguish, in a first step, between lazy and eager techniques (Gray, Helland, O’Neil & Shasha, 1996) and limit attention to considering only the update propagation strategy. Both of these approaches are described hereafter, where additional criteria are considered in order to refine this taxonomy.
EAGER REPLICATION

Eager replication propagates the updates of a transaction before the transaction commits. This ensures the consistency of all replicas, but all of the computing time needed for communication then needs to be added to the transaction lifetime, thus causing response times that are longer than those of lazy replication techniques. However, if an atomic broadcast protocol (Hadzilacos & Toueg, 1993) is used, concurrency control can be dealt with locally, so the overall communication costs can be kept low.

Several eager replication techniques exist. According to Wiesmann et al. (2000), the following three characteristics are used to classify the following techniques.

Server Architecture (Gray et al., 1996)

Considers where updates of a given data item are initially processed. There are two options:

- **Update Everywhere (UE):** The updates can be initially done in any of the replicas of the data item.
- **Primary Copy (PC):** For each data item, a distinguished replica exists, its primary copy. Only the primary copy may initially process an update of such a data item; that is, there is only one active replica, and it is always the same one.

Server Interaction

Analyzes how many messages are exchanged by the servers during transaction processing. There are two alternatives:

- **Linear Interaction (LI):** Servers exchange messages for each operation involved in the transaction. As a result, the number of messages depends on the transaction length.
- **Constant Interaction (CI):** Servers exchange a fixed number of messages. The typical case is one update message at the end of the transaction.

Transaction Termination

Considers the actions needed by the replicas to decide the result of a transaction. This depends on the determinism of the algorithm. Again, two options exist:

- **Voting Termination (VT):** An additional round of messages is needed to decide whether the transaction has to be committed or aborted. The traditional two-phase commit protocol is a typical example of this kind of termination.
- **Non-Voting Termination (NT):** This option is applicable when all replicas may decide locally on the completion of a transaction. To this end, a deterministic and symmetrical distributed algorithm is needed, in general. However, determinism is not needed when transactions do not conflict; that is, two or more transactions accessing disjoint sets of records may be executed or terminated in any order in different replicas.

Combining the three characteristics above yields eight classes of replication protocols:

1. UE-CI-VT: “Update everywhere” propagation, with “constant interaction” and “voting termination”.
2. UE-CI-NT: “Update everywhere” propagation, with “constant interaction” and “non-voting termination”.
3. UE-LI-VT: “Update everywhere” propagation, with “linear interaction” and “voting termination”.
4. UE-LI-NT: “Update everywhere” propagation, with “linear interaction” and “non-voting termination”.
5. PC-CI-VT: “Primary copy” update propagation, with “constant interaction” and “voting termination”.
6. PC-CI-NT: “Primary copy” update propagation, with “constant interaction” and “non-voting termination”.
7. PC-LI-VT: “Primary copy” update propagation, with “linear interaction” and “voting termination”.
8. PC-LI-NT: “Primary copy” update propagation, with “linear interaction” and “non-voting termination”.

For reducing the communication costs of an eager protocol, CI is better than LI. Traditionally, eager protocols mostly used the LI-VT combination, either with UE or with PC. The authors of this classification proposed UE-CI-VT and UE-CI-NT as the best possible alternatives for eager protocols; some examples of them have been presented in Kemme and Alonso (2000). Each of these protocols needs atomic broadcast. The protocol described in Irún, Muñoz, Decker, and Bernabéu (2003) belongs to the UE-CI-VT class and only requires uniform reliable broadcast, which is faster than atomic broadcast. However, since reliable broadcast does not guarantee that all updates are delivered in the same order in all replicas, a voting termination procedure is needed, since each replica cannot determine locally if a transaction has to be committed or aborted. Thus, using reliable broadcast without voting, the UE-CI-NT technique cannot be implemented.

The UE propagation technique is less scalable than the PC approach, since a coordination phase is needed to find out if different transactions collide, while in the PC
Related Content

A Systematic Approach to Flexible Specifications, Composition, and Restructuring of Workflow Activities
[www.igi-global.com/article/systematic-approach-flexible-specifications-composition/3304?camid=4v1a](www.igi-global.com/article/systematic-approach-flexible-specifications-composition/3304?camid=4v1a)

Database Benchmarks
[www.igi-global.com/chapter/database-benchmarks/7967?camid=4v1a](www.igi-global.com/chapter/database-benchmarks/7967?camid=4v1a)

Flexible Querying of Imperfect Temporal Metadata in Spatial Data Infrastructures
[www.igi-global.com/chapter/flexible-querying-imperfect-temporal-metadata/52300?camid=4v1a](www.igi-global.com/chapter/flexible-querying-imperfect-temporal-metadata/52300?camid=4v1a)

Integrity Constraints Checking in a Distributed Database
[www.igi-global.com/chapter/integrity-constraints-checking-distributed-database/44387?camid=4v1a](www.igi-global.com/chapter/integrity-constraints-checking-distributed-database/44387?camid=4v1a)