A Formal Analysis of the Deferred Update Technique*

Rodrigo Schmidt[1,2] and Fernando Pedone[2]

[1] École Polytechnique Fédérale de Lausanne, Switzerland
[2] University of Lugano, Switzerland

Introduction. In the deferred update technique for database replication, a number of database replicas are used to implement a single serializable database interface. Its main idea consists in executing all operations of a transaction initially on a single database replica. Transactions that do not change the database state can commit locally to the replica they executed, but other transactions must be globally certified and, if committed, have their update operations submitted to all replicas. Despite its wide use, we are not aware of any work that explored the inherent limitations and characteristics of deferred update database replication, ours being the first attempt in this direction.

We specify a general abstract deferred update algorithm that embraces all the protocols we know of. This general specification allows for a better understanding of the technique, including its requirements and limitations and can be used to ease designing and proving specific protocols.

The Deferred Update Abstraction. Due to the space limitation, we present only the general idea of our approach and results in this brief announcement. In our extended technical report [1], we present complete specifications and explain the results in detail. We start with a general serializable database specification, later used to prove our abstraction correct, and gradually move towards an abstract deferred update algorithm. All our specifications have been translated into the TLA$^+$ specification language [2] and model checked.

From our initial specification of a serializable database, we formalize the notion of *order-preserving serializability*, introduced by Beeri et al. in the context of nested transactions [3], for its use in deferred update replication. While previous works assumed replicas should satisfy *order-preserving serializability* to ensure global serializability, we show that serializability is guaranteed if replicas satisfy the weaker notion of *active order-preserving serializability* that we introduce. Some multiversion concurrency control mechanisms [4], for example, are active order-preserving but not strict order-preserving; yet, our results show that they can be used in deferred updated protocols.

Our specification of serializability also allows us to reason better about the actions required to implement it using a number of internal database replicas. From that, we could specify the atomic actions that abstract the deferred update technique, that is, the actions implemented by all deferred update protocols. The abstract deferred update algorithm we present allows us to isolate the properties

* The work presented in this paper has been partially funded by the SNSF, Switzerland (project #200021-170824)

of the *termination protocol*, responsible for certifying and propagating update transactions to the replicas. We specify three safety properties, namely *Non-triviality*, *Stability*, and *Consistency*, and discuss their necessity. This discussion brings out the result that update transactions cannot be propagated to replicas in different orders, even if they operate (read or write) on completely disjunct subsets of the data items, for it can break serializability. This is rather counter-intuitive since serializability would allow such transactions to be scheduled in any order. Our properties imply that the termination protocol must ensure something stronger than just the serializability of the update transactions, which means that proving only that does not suffice.

By extending the termination protocol with a simple liveness property that ensures propagation of committed transactions, we were able to show that its properties necessarily implement, for committed transactions, the Sequence Agreement problem explained in [5]. Briefly, in the sequence agreement problem, a set of processes agree on an ever-growing sequence of values, built out of proposed ones. This problem is a sequence-based specification of the celebrated Atomic Broadcast problem. Our result implies that implementations of the termination protocol are free to abort transactions, but they must atomically broadcast the transactions they commit. As a consequence, any lower bound or impossibility result for atomic broadcast and consensus applies to the termination protocol.

Conclusion. We have formalized the deferred update technique for database replication and stated some intrinsic characteristics and limitations of it. Previous works have only considered new algorithms, with independent specifications, analysis, and correctness proofs. To the best of our knowledge, our work is the first effort to formally characterize this family of algorithms and establish its requirements. Our general abstraction can be used to derive other lower bounds as well as to create new algorithms and prove existing ones correct. Some algorithms can be easily proved correct by a refinement mapping to ours. Others may require an additional effort due to their extra assumptions, but the task seems easier than with previous formalisms. In our experience, we have successfully used our abstraction to obtain interesting protocols and correctness proofs.

References

1. Schmidt, R., Pedone, F.: A formal analysis of the deferred update technique. Technical report, EPFL (2007)
2. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
3. Beeri, C., Bernstein, P.A., Goodman, N.: A model for concurrency in nested transaction systems. Journal of the ACM **36**(2) (April 1989) 230–269
4. Bernstein, P., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley (1987)
5. Lamport, L.: Generalized consensus and paxos. Technical Report MSR-TR-2005-33, Microsoft Research (2004)