Rollbacks in Pessimistic Distributed TM—Abstract

Konrad Siek, Paweł T. Wojciechowski
Poznań University of Technology
konrad.siek@cs.put.edu.pl, pawel.t.wojciechowski@cs.put.edu.pl

Transactional Memory (TM) [2] aims to simplify concurrency control while retaining a decent level of efficiency in comparison to low-level mechanisms such as locks. A distributed TM aims to accomplish the same in a distributed environment (e.g., [5, 10]), although this poses new concerns, like presence of a single point of failure, limits on communication and copying data due to possible network saturation, partial failures, distributed deadlock etc.

The typical *modus operandi* of TM is to execute transactions and restart those that conflict with other transactions—the optimistic approach. This presents a problem, when some operations included in transactions are *irrevocable*, like system calls or network messages—these cannot be easily reversed once they are executed and they may cause inconsistencies. This is particularly important to distributed systems. Optimistic transactions cope with this by executing irrevocable transactions one by one [7] or providing multiple versions of transaction view [1, 4]. Instead, as suggested by [3] and [8, 9], it may be useful to turn to a pessimistic approach over the optimistic one. Pessimistic transactions prevent conflicts by e.g., postponing operations on shared objects. Therefore, they do not abort and so, avoid the problems of irrevocability altogether.

Our research explores the concept of pessimistic TM as a feasible solution for distributed concurrency control environments providing good support for irrevocable operations, a high level of parallelism, and resilience to failures. We base our work on versioning algorithms [8, 9] which use versions to defer operations on objects. We concentrate on the Supremum Versioning Algorithm (SVA), which allows fine-grained synchronization, and supports an early release mechanism for shared objects, allowing more transactions to run in parallel.

The aspect of our research we wish to concentrate on is making pessimistic algorithms fully articulated by allowing *rollback*. It is important because programmers can use it to efficiently cancel the effects of transactions without copying objects over the network and without manually equipping remote objects with an ability to make copies on-site. More importantly, some form of rollback is necessary to handle partial failures.

However, rollback support is not inherent in the pessimistic approach in general and versioning algorithms in particular. For one, objects should be reverted when a transaction rolls back even if they were released early. We propose a solution for this problem in [6] by deferring transaction commitment until preceding transactions commit and demonstrate that opacity and strong progressiveness are nevertheless preserved. Alas, in certain scenarios a transaction that releases objects early and rolls back may force any other transaction to also roll back. Then, some transaction containing irrevocable operations may be randomly aborted due to cascading rollback. We currently work on a solution to this problem, which would remove the necessary condition for forced rollbacks by postponing premature object accesses in irrevocable transactions.

The next step in our research is to design fault-tolerance support for pessimistic transactions. This will be possible because rollback, its key ingredient, is already available. Therefore, failing transactions can be aborted, their effects withdrawn, and the objects they used released for use by other transactions. We plan to formally determine the properties of the fault-tolerance component as well as the underlying versioning algorithms. Our further research involves crash-recovery support.

References

[1] H. Attiya and E. Hillel. Single-version STMs can be multi-version permissive. In *Proc. of ICDCN'11*, Jan. 2011.

[2] M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *Proc. of ISCA'93*, May 1993.

[3] A. Matveev and N. Shavit. Towards a fully pessimistic STM model. In *Proc. of TRANSACT '12*, Aug. 2012.

[4] D. Perelman, R. Fan, and I. Keidar. On maintaining multiple versions in STM. In *Proc. of PODC'10*, July 2010.

[5] M. M. Saad and B. Ravindran. Supporting STM in distributed systems: Mechanisms and a Java framework. In *Proc. of TRANSACT '11*, June 2011.

[6] K. Siek and P. T. Wojciechowski. Brief announcement: Towards a fully-articulared distributed transactional memory. In *Proc. of SPAA '13*, June 2013.

[7] A. Welc, B. Saha, and A.-R. Adl-Tabatabai. Irrevocable transactions and their applications. In *Proc. of SPAA '08*, June 2008.

[8] P. T. Wojciechowski. Isolation-only transactions by typing and versioning. In *Proc. of PPDP '05*, July 2005.

[9] P. T. Wojciechowski. *Language Design for Atomicity, Declarative Synchronization, and Dynamic Update in Communicating Systems*. Poznań University of Technology Press, 2007. Habilitation thesis.

[10] P. T. Wojciechowski, T. Kobus, and M. Kokociński. Model-driven comparison of state-machine-based and deferred-update replication schemes. In *Proc. of SRDS '12*, Oct. 2012.