The Impossibility of Ensuring Snapshot Isolation in Genuine Replicated STMs *

Masoud Saeida Ardekani
LIP6-Regal
masoud.saeida-ardekani@lip6.fr

Pierre Sutra
INRIA/LIP6-Regal
pierre.sutra@lip6.fr

Marc Shapiro
INRIA/LIP6-Regal
marc.shapiro@acm.org

Abstract

In order to scale to large multiprocessors or clusters, transactional memories must reduce reliance on synchronisation. Therefore, we consider two favorable formal properties, namely Snapshot Isolation (SI, by which read-only transactions commit without synchronisation), and Genuine Partial Replication (GPR, by which a processor synchronises for a transaction only if it maintains a copy of some data item accessed by that transaction). We show that, unless the read-set of every transaction is known in advance, the combination of SI+GPR is impossible. To circumvent this impossibility result, we propose to weaken SI such that snapshots are allowed to be non-monotonic.

1 Introduction

Software Transactional Memory (STM) is a recent paradigm for concurrent programming by providing the well-known concepts of atomicity, consistency, and isolation properties to the programmer.

Initial work on STM considered mostly small-scale cache-coherent multi-core machines. We are now interested in scaling STMs to distributed memories, such as processors without hardware shared memory or small clusters. Over the past few years, several STMs [1, 2, 3, 4, 5] have been proposed trying to address this issue.

One unique characteristic of STMs, compared to database transactions, is that the cost of synchronisation dominates processing time; this imbalance can only worsen with larger processor architectures. Therefore, scalability requires to minimise the synchronisation requirements.

Running transactions in distributed-memory processor architectures requires replicating the shared data. However, the classical State-Machine Replication (SMR) approach forces a purely sequential execution.[1]

These issues justify our interests in the two following formal properties. *(i)* In Generalized Snapshot Isolation [6] (SI) a transaction runs against a consistent snapshot of the data (not necessarily the most recent version). SI has the advantage that a read-only transaction does not synchronise, as it can commit unilaterally. Furthermore, aborts are reduced, as an update aborts only if it conflicts with a concurrent, already-committed transaction. *(ii)* In Genuine Partial Replication (GPR), a given data item is replicated only on some subset of processors, and a processor synchronises for a transaction only if it maintains a replica of some data item accessed by that transaction. This allows independent transactions to commit in parallel.

In this paper, we report on two early results of this research. One is an impossibility result: the combination of SI with GPR is impossible.[2] The other is a proposal for circumventing the impossibility: to weaken the SI requirement for snapshot monotonicity, i.e., to allow the snapshot of two concurrent read-only transactions to contain different updates.

2 Genuine Replication and SI

Under *snapshot isolation*, a transaction T always sees a consistent state of the system, and if T is a read-only transaction, it always commits.

Genuine replication states that to execute a transaction T over a set of objects O, only processes that hold a copy of an object $o \in O$ may take steps [7]. Therefore, genuineness increases the scalability by parallelizing non-conflicting transactions as much as possible. [3]

We have proved an impossibility result that states no message-passing transactional system can achieve both snapshot isolation and genuine partial replication if data items accessed by each transaction are not known in advance.

Intuitively, this impossibility result stems from the fact that it is impossible to take consistent and monotonic snapshots without violating genuineness.

The most well-known algorithm to take a snapshot in a replication system is to use atomic broadcast, and total order all transactions. Since all transactions are totally ordered, each transaction can take a snapshot by reading the latest committed versions. Our result implies that we cannot modify the above system to obtain a genuine partial replication system. In particular, because data items are not known in advance, we cannot replace the broadcast primitive by atomic multicast.

*This research is supported in part by ANR projects Prose (ANR-09-VERS-007-02) and ConcoRDanT (ANR-10-BLAN 0208).

[1] Unless the data can be strictly partitioned statically.

[2] Unless the read-sets of transactions are known in advance. This may be a reasonable expectation in a database with stored procedures, but the STM programming model is much more dynamic.

[3] In shared-memory systems, this property transposes in disjoint-access parallelism [8].

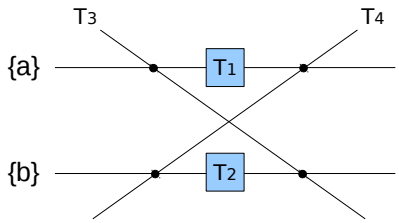

Figure 1: Non-monotonic Snapshots

3 Weak Snapshot Isolation

To circumvent the impossibility result while still preserving scalability (genuiness), and liveness (wait-freedom for read-only transactions), we first propose a novel decomposition of SI into the following properties:

- Consistent Snapshot: this property states that each transaction should read a consistent state of the memory.
- Write Conflict Freedom: this property means that two concurrent transactions do not write on the same object.
- Snapshot Monotonicity: this property states that transactions should observe non-conflicting concurrent transactions in the same order.

To understand the snapshot monotonicity, consider there are two different processes that hold two different objects namely a and b (figure 1). Transactions T_1 and T_2 write on objects a and b respectively, while read-only transactions T_3 and T_4 try to read both a and b. In spite of the fact that both of the snapshots taken by T_3 and T_4 are consistent snapshots, they are non-monotonic snapshots. Thus, the execution in figure 1 violates the SI rules, and cannot be accepted by a replication system that ensures SI.

We observed that while ensuring snapshot monotonicity between different transactions of a same process is crucial (local snapshot monotonicity), ensuring snapshot monotonicity among transactions of different processes (global snapshot monotonicity) is not essential, hence we can relax this property in order to increase the scalability.

Therefore, we propose a novel consistency criteria called weak snapshot isolation (WSI). WSI ensures local snapshot monotonicity along with ensuring consistent snapshot and write conflict freedom. Roughly speaking, WSI provides correctness guarantees very similar to SI. In particular, queries always see a consistent snapshot, and two conflicting updates cannot commit both. The subtle difference with SI lies in the fact that WSI excludes global snapshot monotonicity, and allows concurrent transactions from different processes observe non-conflicting updates in different orders as it is shown in figure 1.

4 Conclusion and Future Works

In this paper, we presented our ongoing research on making STMs scalable by using genuine partial replications along with SI. We presented the impossibility results about genuine partial replication under SI in any transactional system. To circumvent the impossibility result, we first proposed a novel decomposition of SI into three properties namely consistent snapshot, write conflict freedom, and snapshot monotonicity. We then introduced weak snapshot isolation that instead of ensuring snapshot monotonicity only ensures local snapshot monotonicity.

In our ongoing studies, we plan to build a complete genuine replicated STM under WSI, and compare it with other state of the art replicated solutions. We are also investigating the possible similarities between WSI and other consistency criteria introduced so far in the literature.

Moreover, we are trying to prove a conjecture that states no message-passing transactional system can achieve both snapshot isolation and genuine partial replication if a read-only transactions can commit while contacting a *majority* of replicas.

References

[1] Annette Bieniusa and Thomas Fuhrmann. Consistency in hindsight: A fully decentralized STM algorithm. In *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pages 1–12. IEEE, 2010.

[2] Robert L. Bocchino, Vikram S. Adve, and Bradford L. Chamberlain. Software transactional memory for large scale clusters. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming - PPoPP '08*, PPoPP '08, page 247, New York, New York, USA, 2008. ACM Press.

[3] Kaloian Manassiev, Madalin Mihailescu, and Cristiana Amza. Exploiting distributed version concurrency in a transactional memory cluster. In *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming PPoPP 06*, page 198. ACM, ACM Press, 2006.

[4] Nuno Carvalho, Paolo Romano, and Luís Rodrigues. SCert. In *Proceedings of the 4th Annual International Conference on Systems and Storage - SYSTOR '11*, page 1, New York, New York, USA, May 2011. ACM Press.

[5] Maurice Herlihy and Ye Sun. Distributed transactional memory for metric-space networks. *Distributed Computing*, 20(3):195–208, 2007.

[6] S. Elnikety, W. Zwaenepoel, and F. Pedone. Database Replication Using Generalized Snapshot Isolation. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*, pages 73–84. IEEE Computer Society, October 2005.

[7] Nicolas Schiper, Pierre Sutra, and Fernando Pedone. P-Store: Genuine Partial Replication in Wide Area Networks. In *2010 29th IEEE Symposium on Reliable Distributed Systems*, pages 214–224. IEEE, October 2010.

[8] Amos Israeli and Lihu Rappoport. Disjoint-access-parallel implementations of strong shared memory primitives. In *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing - PODC '94*, pages 151–160, New York, New York, USA, August 1994. ACM Press.