Distributed Software Transactional Memories: A Summary of Research @ IST/INESC-ID

Luís Rodrigues and Paolo Romano
INESC-ID, Instituto Superior Técnico, Universidade Técnica de Lisboa

I. Introduction

Distributed Transactional Memory (DTM) aims at introducing a novel programming paradigm combining the simplicity of Transactional Memory (TM)[11] with the scalability and failure resiliency achievable by exploiting the resource redundancy of distributed platforms. These features make the DTM model particularly attractive for inherently distributed application domains such as Cloud computing or High Performance Computing (HPC). In the HPC domain, several specialized programming languages (such as X10 or Fortress) already provide programmatic support for the DTM abstraction. In the context of Cloud computing, several recent NoSQL data-grids platforms expose transactional APIs and rely on in-memory storage of data to achieve higher performance and elasticity.

The Distributed Systems Group at INESC-ID has started to perform research on DTMs in 2008. In that year, we have presented at the LADIS workshop a position exposing a number of research directions that we aimed at addressing[18]. Over the last 4 year we have published a number of papers[21], [20], [6], [19], [7], [3], [13], [5], [4], [22], [8], [16] based on prototypes that we have built to explore these research directions, in an attempt to better understand the advantages and limitations of the DTM paradigm. The current paper makes an overview of this research, highlighting the main findings and providing directions for future work.

II. Research Directions

Our research has started to address the development of fully replicated DTMs offering serializability[2] and opacity[10]. We started by observing that, in DTM systems, transactions are typically must shorter than in other transactional systems, e.g. DBMSs. Therefore, distributed coordination is likely to have a relatively amplified detrimental impact on performance and cause severe CPU underutilization[18]. To address this problem, we have initially explored two lines of research: replication schemes that could *reduce coordination latency* by minimizing the amount of information exchanged among nodes and/or by using efficient communication primitives; and *speculative approaches* that could exploit the idle CPU time induced by distributed coordination.

From the experience with the prototypes we have built it became clear that no solution would excel for all workloads. This prompted a complementary line of research: investigating *autonomic mechanisms* capable of selecting, at runtime and in an automated fashion, the replica synchronization protocol that best fits the current workload characteristics. The next step has been pursuing higher scalability levels, working on the design of *scalable protocols for partial replication*. Finally, the growing interest in distributed key-value stores with different consistency requirements, have also led us to study replication schemes for transactional key-value stores ensuring *weaker consistency criteria*.

III. Research Results

Minimizing Replica Coordination Cost. The first idea we investigated to minimize replica coordination costs consisted in exploiting the space-efficient encoding properties of Bloom filters (BF) to reduce the information exchanged among replicas[6]. By using BFs to encode transaction read-sets, we have shown that is possible to reduce significantly the network traffic (yielding remarkable benefits even in LAN settings) at the expense of a small, user-tunable, increase of the abort rate.

In a second research line, we studied how to leverage on the abstraction of weak mutual exclusion[21], [20], to conceive a lease-based replication protocol that could take advantage of data access locality to avoid using expensive consensus-based coordination primitives. The resulting protocol, which we named ALC (Asynchronous Lease-based Certification) allows reducing coordination latency by up to one order of magnitude, and, interestingly, proved to be useful even in scenarios with high contention and low locality[3].

Speculative replication. Since in most TM workloads the ratio between transaction processing time and replica coordination latency can be extremely small, speculative replication techniques appeared as a feasible solution to amortize coordination costs and/or overlap processing and communication.

A first idea we explored was based on activating multiple speculative instances of a transaction along different serialization orders, in parallel with the execution of the replication protocol, so to enhance the likelihood of committing a speculative instance as soon as the final transaction serialization order is determined. The key challenge consists in identifying, at run-time and in an efficient way, the set of alternative serialization orders that would lead a transaction to observe different snapshots of the TM: a problem that is known to be NP-complete[14], and for which we provided a complete algorithm[19], as well as a lightweight, pragmatical heuristic that trades completeness for efficiency[12].

We studied also an alternative approach that aimed to leverage speculation to overlap processing and communication. Our first algorithms[13], [5] (addressing respectively active replication and certification based replication) were based on

the idea of propagating the snapshots generated by speculatively committed transactions, i.e. transactions whose final outcome is not yet determined by the replica coordination protocol. However, they would blocked speculatively committed transactions till the replication protocol determined their final outcome. This solution straightforwardly prevents transactions from externalizing inconsistent states, but leads to minimal benefits in low-conflict scenarios, in which speculatively committed snapshots are unlikely to be accessed by concurrent transactions. SPECULA[17], addresses this issue by allowing application level threads to commit speculatively a sequence of transactions and detecting automatically, using transparent byte-code injection techniques, whether speculation should be blocked to prevent externalizing inconsistent outputs.

No-one-size-fits-all. Our work with the protocols above highlighted the (non-surprising) fact that no protocol outperforms the others in all workloads. Therefore we have built a framework where different protocols can coexist[4], and have studied techniques to select the most appropriate protocol in runtime. This involved the design of learning techniques to estimate the costs of different coordination primitives as a function of the message size[7]. We have later designed a protocol that, using these online learners, is able to select the most suitable coordination protocol, on a message-by-messages basis[8].

Addressing Partial Replication. All the work referred above considered fully replicated systems, i.e., all nodes maintain a consistent copy of the entire dataset. Motivated by the challenge to achieve higher scalability levels, over the last 2 years we have been working on partial replication protocols. In this context, a key requirement to maximize scalability is to ensure *genuineness*, i.e., to guarantee that only the sites that replicate the data items accessed by a given transaction exchange messages to decide its final outcome.

Our most recent result in this area, GMU[16], is, to the best of our knowledge, the first genuine, and hence highly scalable, multi-versioning protocol supporting invisible reads and wait-free read-only transactions, hence achieving excellent performance in read-dominated workloads, as typical of a wide range of real-world applications. Interestingly, achieving this result required introducing a slight relaxation of classic One Copy Serializability (1CS)[2]: GMU in fact guarantees update-serializability[1], a consistency criterion weaker than 1CS but still compliant with the ANSI SERIALIZABLE isolation level.

Weaker Consistency Models. Finally we have recently started to address the support for weaker consistency models. This work has been performed in the context of the CloudTM european project, where we cooperate with RedHat to find more efficient implementations of the Inifinispan data grid. Our work has shown that the above ideas can be extended to weaker models, outperforming naive (albeit extremely popular) coordination schemes such as distributed locking and two-phase commit[22].

IV. Conclusions and Future Work

We have started research on DTMs considering "traditional" Software Transactional Memory systems, mostly motivated by the fact that we have a real system using this technology in production at IST. However, many of the techniques we have designed can have application in different settings. In fact, techniques such as ALC can be applied to any replicated system. An interesting experience would be to apply this technique in more traditional settings such as classical database replication[15]. Also, the recent work of [9] has shown that similar techniques can be applied inside many-cores, using network-on-chip communication. Finally, the advent of cloud computing has spurred the proliferation of large-scale distributed NoSQL data stores providing different levels of support for transactional operations. These platforms represent a natural, challenging test-bed to apply these results, and motivating further research in this intriguing area.

Acknowledgments: This work was partially supported by the FCT (INESC-ID multi annual funding through the PIDDAC Program fund grant, under project PEst-OE/ EEI/ LA0021/ 2011, and by the project PTDC/EIA-EIA/102212/2008) and also by project CloudTM (FP7-ICT 257784).

References

[1] A. Adya. *Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions*. PhD thesis, MIT, 1999.

[2] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[3] N. Carvalho, P. Romano, and L. Rodrigues. Asynchronous lease-based replication of software transactional memory. In *Middleware*, pages 376–396, 2010.

[4] N. Carvalho, P. Romano, and L. Rodrigues. A generic framework for replicated software transactional memories. In *NCA*, 2011.

[5] N. Carvalho, P. Romano, and L. Rodrigues. Scert: Speculative certification in replicated software transactional memories. In *SYSTOR*, 2011.

[6] M. Couceiro, P. Romano, N. Carvalho, and L. Rodrigues. D2STM: dependable distributed software transactional memory. In *PRDC*, 2009.

[7] M. Couceiro, P. Romano, and L. Rodrigues. A machine learning approach to performance prediction of total order broadcast protocols. In *SASO*, 2010.

[8] M. Couceiro, P. Romano, and L. Rodrigues. Polycert: Polymorphic self-optimizing replication for in-memory transactional grids. In *Middleware*, 2011.

[9] V. Gramoli, R. Guerraoui, and V. Trigonakis. TM2C: a software transactional memory for many-cores. In *EuroSys*, 2012.

[10] R. Guerraoui and M. Kapalka. On the correctness of transactional memory. In *PPoPP*, pages 175–184, 2008.

[11] M. Herlihy and J. Moss. Transactional memory: architectural support for lock-free data structures. In *ISCA*. ACM, 1993.

[12] R. Palmieri, F. Quaglia, and P. Romano. OSARE: Opportunistic speculation in actively replicated transactional systems. In *SRDS*, pages 59–64, 2011.

[13] R. Palmieri, P. Romano, and F. Quaglia. Aggro: Boosting stm replication via aggressively optimistic transaction processing. In *NCA*, 2010.

[14] C. Papadimitriou. The serializability of concurrent database updates. *J. ACM*, 26(4):631–653, 1979.

[15] M. Patiño Martínez, R. Jiménez-Peris, B. Kemme, and G. Alonso. Scalable replication in database clusters. In *DISC*, 2000.

[16] A. Peluso, P. Ruivo, P. Romano, Quaglia F., and L. Rodrigues. When scalability meets consistency: Genuine multiversion update-serializable partial data replication. In *ICDCS*, 2012.

[17] S. Peluso, J. Fernandes, P. Romano, F. Quaglia, and L. Rodrigues. SPECULA: speculative replication of software transactional memory. In *SRDS*, October 2012.

[18] P. Romano, N. Carvalho, and L. Rodrigues. Towards distributed software transactional memory systems. In *LADIS*, 2008.

[19] P. Romano, R. Palmieri, F. Quaglia, N. Carvalho, and L. Rodrigues. An optimal speculative transactional replication protocol. In *ISPA*, Taiwan, Taipei, 2010.

[20] P. Romano and L. Rodrigues. An efficient weak mutual exclusion algorithm. In *ISPDC*, 2009.

[21] P. Romano, L. Rodrigues, and N. Carvalho. The weak mutual exclusion problem. In *IPDPS*, 2009.

[22] P. Ruivo, M. Couceiro, P. Romano, and L. Rodrigues. Exploiting total order multicast in weakly consistent transactional caches. In *PRDC*, 2011.