Distributed Computing Column 45
What Theory for Transactional Memory?

Idit Keidar
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
idish@ee.technion.ac.il

In this issue, we have a short column. It features a review of *WTTM 2011*, the third Workshop on the Theory of Transactional Memory, by Petr Kuznetsov and Srivatsan Ravi. This annual workshop focuses on developing *theory* for understanding transactional memory systems. It discusses recent achievements as well as remaining challenges. Petr and Srivatsan review the most recent instance of the workshop, which was co-located with DISC in September 2011 in Rome.

Many thanks to Petr and Srivatsan for their contribution.

Call for contributions: I welcome suggestions for material to include in this column, including news, reviews, open problems, tutorials and surveys, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways.

WTTM 2011
The Third Workshop on the Theory of Transactional Memory[*]

Petr Kuznetsov Srivatsan Ravi
TU Berlin/Deutsche Telekom Laboratories
petr.kuznetsov@tu-berlin.de ravi@net.t-labs.tu-berlin.de

Transactional Memory (TM) is a programming paradigm which promises to make concurrent programming tractable and efficient. A lot of work is currently done on the design and implementation of TM systems. However, a sound theoretical framework to reason about the TM abstraction is yet to come.

In conjunction with DISC 2011, the TransForm project (Marie Curie Initial Training Network) and EuroTM (COST Action IC1001) supported the 3rd edition of the Workshop on the Theory of Transactional Memory (WTTM 2011). The objective of WTTM was to discuss new theoretical challenges and recent achievements in the area of transactional computing.

The workshop took place on September 22-23, 2011, in Rome, Italy. Below we give highlights of the topics discussed during these two days.

1 Semantics and Models

In the first talk of WTTM3, Michael Scott argued that researchers in the area of concurrent systems and transactional memory have now a rare (and possibly brief!) privilege of "making the semantics right." Michael listed a few points that he believed to be crucial for the semantics of TM: (1) atomicity is central, (2) strong atomicity is a non-issue, (3) small transactions are what matter, and (4) privatization is essential, since it solves the problem of legacy synchronization. Among the research challenges inspired by this view, Michael mentioned handling non-transactional memory accesses, long transactions, and understanding the relationship between atomicity and determinism.

Maurice Herlihy presented a *periodic table of progress conditions* that unifies various progress conditions, from lock-based ones, such as deadlock- and starvation-freedom, to non-blocking (lock-free) ones, such as obstruction-freedom and wait-freedom. This is a joint work with Nir Shavit, more detail in the forthcoming paper [11].

[*]The workshop has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement N 238639, ITN project TRANSFORM and EuroTM (COST Action IC1001).

In a somewhat provocative talk, Nir Shavit proposed to reconsider the traditional *optimistic* approach in STM design, that allows a transaction to execute speculatively with an option to abort when run into inconsistencies. It is believed that the optimistic approach gives significant performance benefits. Surprisingly, however, Nir showed that a simple fully pessimistic STM in which no transaction ever aborts may exhibit performance that is comparable to the most efficient optimistic STMs. As a bonus, the proposed pessimistic STM also provides full privatization. This is a joint work with Alex Matveev.

STM is often believed to be an application of techniques devised in the database community to concurrent systems. Is this a right perspective? Or is the very term *transaction* a misnomer for an STM operation? Sandeep Hans presented a comparison on STM and DB consistency conditions, highlighting important similarities and differences. This is a joint work with Hagit Attiya.

2 Synchronization Techniques

In the context of *blocking* (i.e., lock-based) linearizable implementations of concurrent data structures, Panagiota Fatourou proposed to reconsider the recently proposed *combining* approach [10]. Roughly, the approach assumes a single thread, called the combiner, holding a coarse-grain lock on the implemented object and serving requests announced by other (locally spinning) threads. Panagiota described two extensions of this idea optimized for models with coherent caches and cache-less Non-Uniform Memory Access (NUMA) models. Experiments show that the resulting implementations outperform all previous state-of-the-art combining based and fine-grain synchronization algorithms. This is a joint work with Nikolaos D. Kallimanis, more detail in the forthcoming PPoPP paper [9].

Idit Keidar discussed the *locality* aspects in STM running on NUMA multi-cores. She reported that a locality-conscious approach for maintaining versioned locks in TL2 [7] resulted in double speedup on STAMP benchmark. This speedup can be explained by the following factors: 1) improved spacial and temporal locality, 2) reduced false sharing and 3) lower false conflicts. This is a joint work with Elad Gidron and Dmitri Perelman.

3 Universal Constructions and Transaction-Friendliness

STM can be seen as a *universal data structure* that can be accessed with abstract *operations*, i.e., arbitrary sequences of reads and writes. In case of a conflict (however defined), an operation may abort and leave the structure untouched, but all successful operations should constitute a linearizable history. Faith Ellen presented a construction that employs this approach to transform any sequential data structure into a concurrent one. The resulting implementation is efficient and allows for a high degree of parallelism. This is a joint work with Phong Chuong and Vijaya Ramachandran, extending the recent SPAA paper [5].

The efficiency of STM-based concurrent implementation may depend on the data structure being implemented. Tyler Crain proposed a design for a *transaction-friendly* binary search tree that allows for highly efficient STM-based implementations. The trick is to transiently trade off balanced structural invariants for efficiency. This is a joint work with Vincent Gramoli and Michel Raynal, more detail in the forthcoming PPoPP paper [6].

4 Disjoint-Access-Parallelism, Snapshot Isolation, and Nesting

It is often believed desirable for a TM to allow operations accessing disjoint sets of locations to progress concurrently. The property called *disjoint-access-parallelism* captures this by requiring that different operations do not access the same base object unless they operate on some common part of the data structure. Is this property compatible with the ability of individual operations to make progress independently of each other (*wait-freedom*)? Alessia Milani presented a taxonomy of conventional definitions of disjoint-access-parallelism, showed that all these definitions are provably incompatible with wait-freedom, and proposed a new definition that resolves this issue. This is a joint work with Faith Ellen, Panagiota Fatourou, Eleftherios Kosmas, and Corentin Travers.

Masoud Saeida Ardekani looked at the popular consistency criterion proposed for message-based STMs, called *snapshot isolation* (SI). It turns out that SI cannot be combined with another desirable property, Genuine Partial Replication (GPR). To circumvent this impossibility result, a weaker consistency criterion is proposed that trades off monotonicity of snapshots for efficiency. This is a joint work with Pierre Sutra and Marc Shapiro.

An important STM functionality, called *nesting*, allows a transaction to invoke other transactions. But how do we specify correctness for an aborted nested transaction? What do we mean when we say that two nested transactions conflict? Sathya Peri proposed a new conflict notion nbConf (non-blocking conflict) that facilitates non blocking implementation of the read operation for nested transactions. Using this conflict notion does not cause a scheduler to block, in contrast to the previous conflict notion for nested transactions. The definition of a conflict is generic enough so that it can be used with other correctness criteria as well. This is a joint work with Krishnamurthy Vidyasankar.

5 Contention Management, Virtual Machines, and Speculation

The logic for resolving conflicts between concurrent transactions (*contention management*) is crucial. Danny Hendler gave a survey on *scheduling-based* contention management techniques [15, 8, 1] that intend to improve throughput under high contention.

Annette Bienussa summarized her recent work [2, 3] on virtual machines for clusters of many-core processors or networks of embedded processors. The consistency model in the proposed virtual machine is transactional memory, but it offers backwards compatibility and integration with classical Java synchronization techniques. To this end, she proposed a methodology for handling monitors and volatiles transactionally, yet following the Java memory specification [4]. This is a joint work with Thomas Fuhrmann.

Paolo Romano proposed *speculation* as an approach to boost performance of STM-based replication [14]. He presented a survey of speculative STM replication protocols that have been published so far [12, 13], illustrating how the existing solutions fit in the whole design space of speculative replication protocols, and identifying which regions of this design space are currently unexplored in order to stimulate further research in this challenging area. This is a joint work with Roberto Palmieri, Francesco Quaglia, Nuno Carvalho and Luís Rodrigues.

6 Practical TM Specifications and Verification

Torvald Riegel presented a draft specification that was recently proposed by contributors from several companies (IBM, Intel, Oracle, as well as HP and Red Hat) to equip C++ with transaction statements. He gave an overview of the new transactional language constructs and discussed how transactions integrate with the C++0x memory model, and why it makes sense to specify C++ transactions based on this model.

Maged Michael noticed that, though non-blocking progress was an essential requirement of the earliest proposals for transactional memory systems, almost all recent TM implementations do not support non-blocking transactions. Michael described the challenges of incorporating non-blocking transactions in a practical TM specification in combination with other useful features, with a focus on the aforementioned draft specification for transactional language constructs for C++.

Victor Luchangco summarized the (unique so far!) experience he gained in formal machine-checked verification of a real transactional memory algorithm. The process involved precise formal specifications for transactional memory, precise formal models of TM algorithms, and rigorous proofs that algorithms meet specifications.

7 Discussion: What Are the Coming Challenges?

The workshop gave rise to long discussions on the future of TM research. The idea of using transactional memory for tractable concurrency has been actively entertained by the research community for more than eight years now. However, TM has not yet become a mainstream programming paradigm. Moreover, there is still no convergence on the very foundations of TM models. So how can we change the situation? Below we sketch some important points of these discussions (in the chronological order):

- Michael Scott: there are many interesting problems, but we should not lose the original goal of the TM idea: make life of a mainstream programmer simpler.

- Victor Luchangco: we should gain better understanding of how to specify *relaxed* transactions (e.g., in C++).

- Torvald Riegel: it is good to have precise specifications, but we also need simple *explanations*. Without them mainstream programmers will never adopt the model.

- Panagiota Fatourou: perhaps STM is a bit oversold? We are still unclear which properties of a TM system are the most important. Is it simplicity? Performance gains? Composability?

- Lisa Higham: what is the consistency model of TM? What is the consistency model of a machine on which it runs? The model should be simple (otherwise do not call it transactions!), and it should avoid *double* synchronization: one in the TM and the other in the underlying memory.

- Maurice Herlihy: it is crucial at this point to formulate formal programming language semantics for the transactional concurrency model and determine the class of applications that would be benefitted from the transactional concurrency model.

- Nir Shavit: we can expect modifications in processors to support running small scale transactions as well as support for optimistic multi-word CAS primitives. The question then is, how to design

STM's to backup the hardware transactional primitives when larger transactions are issued and efficiently utilize such hardware features for implementing relaxed transactions. Consequently, this raises the question of how to reason about performance of such hybrid TMs.

- Victor Luchangco: it is unlikely that we shall ever have a *fully* transactional system and the success of TM will depend on our understanding the interaction between transactional and non-transactional code. *Privatization* will hence be crucial.

  Victor also emphasized the need to develop transaction-friendly data structures (*á la* binary search tree implementation described by Tyler).

- Michael Scott: the compiler issues relating to STMs are important, specifically on the amount of state required to rollback and *program slicing*.

- Torvald Riegel: how do we specify the progress conditions provided by practical TM specifications, and how do they depend on the guarantees provided by the OS scheduler (along the lines of [11]).

- Rachid Guerraoui: theoretical research in transactional memory must be treated as a first-class citizen. In particular, intellectually challenging TM-related question should not be discarded purely on the *relevance* basis. After all, the discussion shows that there is no consensus on the very notion of relevance here. Torvald and Victor agreed but insisted that, among interesting theoretical questions, we should prioritize those that are believed to be important to the engineers.

Abstracts and slides of the talks can be found at `http://transform.t-labs.tu-berlin.de/tw11/`.

References

[1] M. Ansari, M. Luján, C. Kotselidis, K. Jarvis, C. C. Kirkham, and I. Watson. Steal-on-abort: Improving transactional memory performance through dynamic transaction reordering. In *HiPEAC*, pages 4–18, 2009.

[2] A. Bieniusa, J. Eickhold, and T. Fuhrmann. The architecture of the DecentVM: towards a decentralized virtual machine for many-core computing. In *Virtual Machines and Intermediate Languages*, VMIL '10, pages 5:1–5:10, New York, NY, USA, 2010. ACM.

[3] A. Bieniusa and T. Fuhrmann. Consistency in hindsight: A fully decentralized stm algorithm. In *Proceedings of the 2010 IEEE International Symposium on Parallel Distributed Processing*, IPDPS 2010, pages 1–12, 2010.

[4] A. Bieniusa and T. Fuhrmann. Lifting the barriers: Reducing latencies with transparent transactional memory. In *Proceedings of the 13th international conference on Distributed computing and networking*, ICDCN'12, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] P. Chuong, F. Ellen, and V. Ramachandran. A universal construction for wait-free transaction friendly data structures. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, SPAA '10, pages 335–344, New York, NY, USA, 2010. ACM.

[6] T. Crain, V. Gramoli, and M. Raynal. A speculation-friendly binary search tree. In *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'12)*, 2012.

[7] D. Dice, O. Shalev, and N. Shavit. Transactional locking II. In *In Proc. of the 20th Intl. Symp. on Distributed Computing*, 2006.

[8] S. Dolev, D. Hendler, and A. Suissa. Car-stm: scheduling-based collision avoidance and resolution for software transactional memory. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, PODC '08, pages 125–134, 2008.

[9] P. Fatourou and N. D. Kallimanis. Revisiting the combining synchronization technique. In *PPoPP (to appear)*, 2012.

[10] D. Hendler, I. Incze, N. Shavit, and M. Tzafrir. Flat combining and the synchronization-parallelism tradeoff. In *SPAA*, pages 355–364, 2010.

[11] M. Herlihy and N. Shavit. On the nature of progress. In Proceedings of the 15th International Conference on Principles of Distributed Systems*, OPODIS '1*, 2011.

[12] R. Palmieri, F. Quaglia, and P. Romano. Aggro: Boosting stm replication via aggressively optimistic transaction processing. In *NCA*, pages 20–27, 2010.

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

[14] P. Romano, R. Palmieri, F. Quaglia, N. Carvalho, and L. Rodrigues. Brief announcement: on speculative replication of transactional systems. In *SPAA*, pages 69–71, 2010.

[15] R. M. Yoo and H.-H. S. Lee. Adaptive transaction scheduling for transactional memory systems. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, SPAA '08, pages 169–178, 2008.