The Distributed Computing Column

BY

Panagiota Fatourou

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

6^{th} Workshop on Theory of Transactional Memory

Maria Couceiro*, Vincent Gramoli† and Paolo Romano*
* University of Lisbon/INESC-ID Portugal
† NICTA and University of Sydney Australia

Abstract

This year, the 6th edition of the Workshop on Theory of Transactional Memory (WTTM) was collocated with PODC 2014 in Paris, and took place on July 14. The objective of WTTM was to discuss new theoretical challenges and recent achievements in the area of transactional computing.

Among the various recent developments in the area of Transactional Memory (TM), one of the most relevant was the support for Hardware TM (HTM), which was introduced in various commercial processors. Unsurprisingly, the recent advent of HTM in commercial CPUs has had a major impact also in the program of this edition of WTTM, which has gathered several works addressing issues related to the programmability, efficiency, and correctness of HTM-based systems, as well as hybrid solutions combining software and hardware TM implementations (HyTM).

As in its previous editions, WTTM could count on the generous support of the EuroTM COST Action (IC1001), and on a set of outstanding keynote talks which were delivered by some of the leading researchers in the area, namely Idit Keidar, Shlomi Dolev, Maged Michael and Michael Scott, who were invited to present their latest achievements.

This edition was dedicated to the 60th birthday of Maurice Herlihy and to his foundational work on Transactional Memory, which was commemorated by Michael Scott in the concluding talk of the event.

This report is intended to give the highlights of the problems discussed during the workshop.

Transactional Memory (TM) is a concurrency control mechanism for synchronizing concurrent accesses to shared memory by different threads. It has been proposed as an alternative to lock-based synchronization to simplify concurrent programming while exhibiting good performance. The sequential code is encapsulated in transactions, which are sequences of accesses to shared or local variables that should be executed atomically. A transaction ends either by committing,

in which case all of its updates take effect, or by aborting, in which case all its updates are discarded.

1 TM Correctness and Universal Constructions

Idit Keidar opened the workshop with a talk presenting a joint work with Kfir Lev-Ari and Gregory Chockler on the characterization of correctness for shared data structures. The idea pursued in this work is to replace the classic and overly conservative read-set validation technique (which checks that all read variables have not changed since they were first read) with the verification of abstract conditions over the shared variables, called *base conditions*. Reading values that satisfy some base condition at every point in time implies correctness of read-only operations. The resulting correctness guarantee, however, is found not to be equivalent to linearizability, and can be captured through two new conditions: *validity* and *regularity*. The former requires that a read-only operation never reaches a state unreachable in a sequential execution; the latter generalizes Lamport's notion of regularity [17] for arbitrary data structures. An extended version of the work presented at WTTM has appeared also in the last edition of DISC [18].

Claire Capdevielle presented her joint work with Colette Johnen and Alessia Milani on solo-fast universal constructions for deterministic abortable objects, which are objects that ensure that, if several processes contend to operate on it, a special *abort* response may be returned. Such a response indicates that the operation failed and guarantees that an aborted operation does not take effect [13]. Operations that do not abort return a response which is legal with respect to the sequential specification of the object. The work presented uses only read/write registers when there is no contention and stronger synchronization primitives, e.g., CAS, when contention occurs [3]. They propose a construction with a lightweight helping mechanism that applies to objects that can return an abort event to indicate the failure of an operation.

Sandeep Hans presented a joint work with Hagit Attiya, Alexey Gotsman, and Noam Rinetzky on an evaluation of TMS1 as a consistency criterion necessary and sufficient for the case where local variables are rolled-back upon transaction aborts [2]. The authors claim that TMS [9] is not trivially formulated. In particular, this formulation allows aborted and live transactions to have different views of the system state. Their proof reveals some natural, but subtle, assumptions on the TM required for the equivalence result.

2 Hardware TM

Maged Michael presented an overview of the HTM implementation in the recent IBM Power8 processor, as well as use cases and performance figures. The TM implementation shipped with the processor integrates several interesting TM features, including the ability to suspend transactions (which is expected to allow for much more efficient hybrid TM designs) and to execute the, so called, Rollback Only Transactions (ROT) (i.e. special transactions intended for single thread speculation that avoid the costs of conflict detection).

In order to synchronize with transactions executing on the, so called, fall-back path, hardware transactions subscribe, upon their start, to the lock that is also used to serialize transactions in the fall-back path. The lazy subscription [7] technique consists of postponing the check that the lock is not held, also called "subscription", from the start phase to the pre-commit phase of the transaction. Tim Harris presented his joint work with Dave Dice, Alex Kogan, Yossi Lev and Mark Moir [8] and listed several pitfalls induced by lazy subscription. They show that this technique is not safe for Transactional Lock Elision [19] because unmodified critical sections executing before subscribing to the lock may behave incorrectly in a number of subtle ways[1]. They also show that recently proposed compiler-based mechanisms for lazy subscriptions [5] are not sufficient to avoid all of the pitfalls identified by their work. Also, they argue that extending such compiler supports to avoid some of the identified pitfalls would add substantial complexity and would usually limit the extent to which subscription can be deferred, hence undermining the effectiveness of the optimization.

Alex Matveev, in a joint work with Yehuda Afek and Nir Shavit, proposed a novel reduced hardware lock elision algorithm (RH-LE) providing a safe (and opaque) concurrency between hardware transactions and their lock-based fallback path. The core idea behind the RH-LE approach is to execute the lock fallback path as a Rollback-Only Transaction (ROT). ROTs are a special kind of hardware transactions introduced in Power8 processors, which track only the memory writes and do not undergo conflict detection. By encapsulating the fall-back in a ROT, one can hide the memory writes issued in the fall-back till its successful commit. This allows hardware transactions to use lazy subscription without exposing them to data inconsistencies. Besides preserving opacity of concurrent hardware transactions, this technique also allows read-only hardware transactions to commit even when there is a concurrent fallback execution.

Hillel Avni presented a work entitled "Evaluating the Addition of Non-Transactional Loads to HTM" which advocates the introduction of load instruc-

[1] `https://blogs.oracle.com/dave/entry/a_simple_lazy_subscription_pathology`

tions (called Non-Transactional Loads, or, shortly, NTL) that are invisible to the transactional system, even if triggered from within a transaction. The talk illustrated the benefits associated with the introduction of NTLs operations in HTM systems, including the possibility of designing more efficient hybrid TM systems [7] and supporting composable COP [1] operations. The presentation also addressed some of the key difficulties underlying the implementation of NTLs, and discussed possible ways to tackle them.

Dan Alistarh, in his joint work with Justin Kopinsky, Petr Kuznetsov, Srivatsan Ravi and Nir Shavit, proposes the first model for hybrid TM systems that formally captures the notion of cached accesses provided by hardware transactions, and precisely defines instrumentation costs in a quantifiable way. The proposed model allows for identifying an inherent trade-off between the degree of concurrency a Hybrid TM (HyTM) implementation provides between hardware and software transactions and the amount of instrumentation overhead the implementation must incur. Several lower bounds on the instrumentation costs of HyTM implementations are derived, and it is shown that the cost of avoiding linear instrumentation overheads (as for instance in HybridNOrec [7]) for progressive implementations is that hardware transactions may be aborted by non-conflicting software transactions.

3 Transaction Semantics and Performance

Michael Scott's keynote, entitled "Transactional Semantics with Zombies", addresses the definition of a formal model of zombie executions. The talk focused on the run-time level, where the semantics of individual operations, such as start, read, write, try-commit, govern the interactions between the compiler and the TM system. For sandboxing TM systems, which allow a doomed (or "zombie") transaction to continue for some time beyond an inconsistent read, run-time level semantics cannot be captured by opacity as currently defined.

Konrad Siek presented his work with Pawel T. Wojciechowski on "Zen and the Art of Concurrency Control". Their work explores the TM safety property space: serializability, opacity, virtual world consistency, and the TMS family, and considers whether they support early release and to what extent. In their presentation, Siek specified how serializability can be combined with some database properties to create a broader spectrum of useful early release supporting TM safety properties. They filled the remaining gap by proposing *Last-use consistency*, a consistency property that excludes those inconsistent views [21].

Shlomi Dolev presented two scheduling results. The first idea is CAR-STM [10] and is used to schedule conflicting transactions on the same threads to avoid further conflicts. The second idea is SemanticTM [4] and the combina-

tion of the two techniques aims at enhancing the robustness of the TM system in presence of conflict prone workloads.

Vincent Gramoli presented his joint work with Petr Kuznetsov and Srivatsan Ravi on the importance of concurrency as a complementary metric to performance. Despite the inherent differences of existing synchronization techniques provided by chip multiprocessors, including locks or hardware transactional memory, one can reason on the best suited synchronization to maximize the concurrency of an application [11]. Performance is still the final desirable goal, yet a concurrency metric helps reasoning in terms of potential performance capabilities regardless of low-level hardware artifacts. They argue that, as the trend is to increase the number of cores, only programs that are highly concurrent will scale with the growing core count. In particular, a program that reaches optimal concurrency (without a too large overhead) should intuitively scale on future chips embedding more cores.

4 Integrating Conditional Variables and Replication in TM

Yujie Liu presented his work with Chao Wang and Michael Spear on "A New API For Transactional Condition Synchronization". In this talk, a new mechanism for transactional condition synchronization was introduced, which leverages explicit predicates specified by the programmer in the form of lambda expressions. The published lambda expressions are evaluated after every transaction commit, by encapsulating their evaluation in an independent transaction (possibly executing in hardware). Unlike prior work, the proposed solution does not require splitting atomic transactions that wait on a condition [20] and is also compatible with hardware and hybrid TM [14].

Maciej Kokocinski presented his work with Tadeusz Kobus and Pawel T. Wojciechowski on the "Safety of Replicated Transactional Memory". In their talk, a distributed variant of TM is considered in which transactional memory is consistently replicated on network nodes for greater availability and fault-tolerance [16, 6]. They argue that opacity [12], a standard TM safety property, is misused when applied to replicated transactional systems. The authors also sketch the requirements for a new safety property that can work well with all kinds of transactional systems, including replicated TM.

5 Conclusion

Transactional memory has finally been integrated in commodity hardware and various questions arose on the best use of these features. While more work is needed to tune and fully exploit them [15], HTMs are here to stay. It is now time to explore how these features can simplify and boost concurrent programming tasks without hampering consistency.

Acknowledgements We are grateful to the speakers, to the program committee members of WTTM 2014 for their help in reviewing this year's submissions and to Panagiota Fatourou for her help in the organization of the event. This event was partially supported by the COST Action IC1001 Euro-TM (`http://www.eurotm.org`).

References

[1] Yehuda Afek, Hillel Avni, and Nir Shavit. Towards consistency oblivious programming. In *Proceedings of the International Conference on Principles of Distributed Systems*, OPODIS'11. Springer, 2011.

[2] Hagit Attiya, Alexey Gotsman, Sandeep Hans, and Noam Rinetzky. Safety of live transactions in transactional memory: TMS is necessary and sufficient. In *Proceedings of the 28th International Conference on Distributed Computing*, DISC'14. Springer-Verlag, 2014.

[3] Hagit Attiya, Rachid Guerraoui, and Petr Kouznetsov. Computing with reads and writes in the absence of step contention. In *Proceedings of the 19th International Conference on Distributed Computing*, DISC'05. Springer-Verlag, 2005.

[4] Hillel Avni, Shlomi Dolev, Panagiota Fatourou, and Eleftherios Kosmas. Abort free semantictm by dependency aware scheduling of transactional instructions. In *Proceedings of the Second International Conference on Networked Systems*, NETYS'14. Springer-Verlag, 2014.

[5] Irina Calciu, Tatiana Shpeisman, and Maurice Herlihy. Improved single global lock fallback for best-effort hardware transa tional memory. In *Transact*. ACM, 2014.

[6] Maria Couceiro, Paolo Romano, Nuno Carvalho, and Luis Rodrigues. D2STM: Dependable distributed software transactional memory. In *Pro-*

ceedings of the 15th Pacific Rim International Symposium on Dependable Computing, PRDC'09. IEEE Computer Society, 2009.

[7] Luke Dalessandro, François Carouge, Sean White, Yossi Lev, Mark Moir, Michael L. Scott, and Michael F. Spear. Hybrid NOrec: A case study in the effectiveness of best effort hardware transactional memory. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS'11. ACM, 2011.

[8] Dave Dice, Timothy L. Harris, Alex Kogan, Yossi Lev, and Mark Moir. Extending hardware transactional memory to overcome the pitfalls of lazy subscription. In *http://labs.oracle.com/scalable*, 2014.

[9] Simon Doherty, Lindsay Groves, Victor Luchangco, and Mark Moir. Towards formally specifying and verifying transactional memory. *Electron. Notes Theor. Comput. Sci.*, 259, 2009.

[10] Shlomi Dolev, Danny Hendler, and Adi Suissa. CAR-STM: Scheduling-based collision avoidance and resolution for software transactional memory. In *Proceedings of the 2008 ACM Symposium on Principles of Distributed Computing*, PODC'08. ACM, 2008.

[11] Vincent Gramoli, Srivatsan Ravi, and Petr Kuznetsov. Brief announcement: From sequential to concurrent: correctness and relative efficiency. In *Proceedings of the 31th ACM Symposium on Principles of Distributed Computing*, PODC'12. ACM, 2012.

[12] Rachid Guerraoui and Michal Kapalka. On the correctness of transactional memory. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '08. ACM, 2008.

[13] Vassos Hadzilacos and Sam Toueg. On deterministic abortable objects. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13. ACM, 2013.

[14] Tim Harris, Simon Marlow, Simon Peyton-Jones, and Maurice Herlihy. Composable memory transactions. In *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '05. ACM, 2005.

[15] Intel. Intel xeon processor e3-1200 v3 product family - specification update, 2014. http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e3-1200v3-spec-update.pdf.

[16] Tadeusz Kobus, Maciej Kokocinski, and Pawel T. Wojciechowski. Hybrid replication: State-machine-based and deferred-update replication schemes combined. In *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems*, ICDCS '13. IEEE Computer Society, 2013.

[17] Leslie Lamport. On interprocess communication. *Distributed Computing*, 1(2), 1986.

[18] Kfir Lev-Ari, Gregory Chockler, and Idit Keidar. On correctness of data structures under reads-write concurrency. In *Proceedings of the International Symposium on Distributed Computing*, DISC'14. Springer-Verlag, 2014.

[19] Ravi Rajwar and James R. Goodman. Speculative lock elision: Enabling highly concurrent multithreaded execution. In *Proceedings of the 34th Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO'01. IEEE Computer Society, 2001.

[20] Michael F. Ringenburg and Dan Grossman. AtomCaml: First-class atomicity via rollback. In *Proceedings of the Tenth ACM SIGPLAN International Conference on Functional Programming*, ICFP '05. ACM, 2005.

[21] Konrad Siek and Pawel T. Wojciechowski. Brief announcement: Relaxing opacity in pessimistic transactional memory. In *Proceedings of the 28th International Symposium on Distributed Computing*, DISC'14. Springer-Verlag, 2014.