Where does Transactional Memory research stand and what challenges lie ahead?

WTM 2012, EuroTM Workshop on Transactional Memory

Maria Couceiro and Paolo Romano
IST/INESC-ID, Lisbon, Portugal

Transactional Memory (TM) is a promising technology that aims to simplify parallel programming by providing a programmer friendly alternative to traditional lock-based concurrency. The past ten years have seen intense research work on software and hardware TM proposals, and has recently led to the first hardware TM implementation for a commodity high-performance microprocessor, and to the inclusion of TM support in the world's leading open source compiler.

EuroTM (COST Action IC1001), in collaboration with the CloudTM project[1], organized the second edition of the EuroTM Workshop on Transactional Memory (WTM 2012). The objective of WTM was to discuss new developments for this era of maturing TM research.

The workshop took place on April 10, in Bern, Switzerland, in conjunction with Eurosys 2012. Below we give highlights on the topics discussed in the workshop.

1 Theoretical Foundations and Algorithms

In the first talk of this session, Serdar Tasiran addressed the problem of how to model and verify programs that use TM supporting relaxed conflict detection mechanisms. In order to maximize performance, several TM implementations included mechanisms allowing the programmer to specify whether TM should ignore certain conflicts. Serdar presented a method for modelling and verifying programs using TM with relaxed conflict detection that allows the programmer to reason sequentially on an abstracted version of the transaction.

Jons-Tobias Wamhoff's presentation addressed the overheads of STMs due to transaction management and instrumentation, highlighting how these can lead to severe performance degradation compared to sequential execution especially when few threads are used. He proposed FastLane [27], a new STM system that differentiates between two types of threads: one thread (the pessimistic master) is allowed to commit transactions without aborting, thus with minimal instrumentation and management costs, while other threads (the speculative helpers) can commit transactions only when they do not conflict.

Vincent Gramoli's presentation focused on how to construct re-usable Concurrent Data Types (i.e. abstract data types that export concurrent methods) that can be easily extended through inheritance, by overloading or adding new methods, or via composition. He presented the Polymorphic Transaction methodology [15], a methodology that allows synthesizing re-usable concurrent data types by exploiting a pre-existing polymorphic set of transactional wrappers (compatible with each other) that preserve the atomicity and concurrency of methods.

The two following presentations focused on efficient distributed consistency algorithms in both fully replicated and partially replicated scenarios.

OSARE [21], presented by Roberto Palmieri, is a recent transactional replication protocol, which aims at amortizing the overhead of replica coordination by speculatively exploring multiple transaction serialization orders in parallel with the execution of the replica coordination protocol. OSARE is particularly attractive in TM settings, in which the latency of replica coordination is often several orders of magnitudes larger than transaction's processing time [25, 22].

Sebastiano Peluso presented GMU [23], a partial replication protocol that achieves high scalability (by guaranteeing genuineness [17]) by ensuring Update Serializability (US) [1], a consistency semantics that is slightly weaker than classic one-copy serializability (1CS) [5]. US ensures a semantics equivalent to 1CS on the history of update transactions (hence ensuring consistent state updates), but allows read-only transactions to observe the updates performed by non-conflicting transactions in different orders (which is not admissible in 1CS). The core of GMU is a distributed multiversion concurrency control algorithm, which relies on a vector clock based synchronization mechanism to track, in a decentralized

[1]FP7 STREP Project Cloud-TM, contract no. 257784, `http://www.cloudtm.eu`

(and consequently scalable) way, both data and causal dependency relations among transactions.

2 Hardware Transactional Memory and Scheduling

Philipp Kirchhofer presented a monitoring infrastructure for HTM (Hardware TM) which generates zero runtime overhead. It consists of hardware units attached to each processor core that monitor state changes, generate corresponding events, and send events to a dedicated memory area of the host system, for later post-processing and visualization.

The following presentation dealt with how TM could be adapted to meet Quality of Service requirements in soft real time systems [20]. Walther Maldonado proposed a state-based approach supporting different execution modes with varying scalability/predictability trade-offs that are dynamically selected to minimize the probability of deadline misses, without however compromising fairness.

Atomic RMI [28], presented by Pawel Wojciechowski, is a concurrency control library that allows programmers to group a sequence of remote method invocations and execute them as a distributed transaction. Atomic RMI uses a pessimistic concurrency control algorithm that dynamically schedules object calls by relying on knowledge about the transactions' access patterns extracted using an inter-procedural data flow and region-based analysis.

Hugo Rito showed how memoization [24] could be incorporated into a Software TM (STM) to runtime to decrease the time spent to re-execute transactions upon their aborts. The proposed approach consists of extending STM transactions with a per-transaction memo-cache, which stores information about methods executed inside the transaction and can be used to accelerate the execution of valid memoized methods.

Nuno Diegues's presentation advocated the usage of parallel nesting of transactions and transaction scheduling as a solution for enhancing the performance of write intensive, conflict prone applications [11]. He presented the results obtained by extending a multi-version, lock-free, STM to incorporate parallel nesting and transaction scheduling, which highlighted that these mechanisms can boost performance in highly-contending, write-dominated workloads, by exploring the latent parallelism within top-level transactions.

3 Language Integration and Tools

In the first talk of the session, Oscar Plata presented a mechanism that takes advantage of TM's optimistic concurrency control in order to parallelize iterative computations when no data dependence information is available before runtime. Transactions are validated using a distributed commit protocol and through an eager (conflict detection) - lazy (data versioning) model. This approach has been implemented in a light-weight STM, TinySTM, and preliminary results using synthetic benchmarks show a significant reduction of the number of transaction aborts.

Chris Seaton discussed the idea of combining TM with dataflow in order to leverage on the benefit of both approaches [26]. In the proposed approach TM can be used to make the access to shared state atomic and compatible with the dataflow model of implicit parallelism and scheduling. To explore this idea, the Scala language was extended with constructs for dataflow programming and support for TM. By applying this approach to the Lee's algorithm for circuit routing [3], it is possible to demonstrate not only improvements in programability, in terms of the required volume of code and number of operations related to parallelism, but also performance gains over implementations relying both on coarse-grained locking and only TM.

Fernando Carvalho presented an extension of his previous work on adaptive object metadata [7] that enables switching between two meta-data representation modes: a compact meta-data representation mode, optimized for low contention scenarios, as it minimizes memory overheads but may suffer of aliasing phenomena leading to unnecessary aborts; an extended meta-data representation mode, which has opposite advantages/drawbacks when compared to the former meta-data representation technique, thus resulting tailored for high contention workloads. Preliminary results, obtained by integrating this adaptive layout mechanism in the lock-free version of the JVSTM [13] have show an improvement in the performance and a reduction in the memory overheads for workloads where the number of objects written is much lower than the total number of transactional objects.

Also the following talk, by Ricardo Dias, addressed the issue of how to maximize the efficiency of a TM by optimizing the memory layout of the meta-data associated with transactional objects. Specifically, the talk described the adaptation and extension of DeuceSTM [19] to support an in-place metadata strategy in an efficient and trasparent fashion.

Ricardo Filipe also discussed the issue of parallel nesting in STMs, a feature that is still lacking in most of existing STMs. This limitation, it is argued, hinders the possibility of developing highly parallel TM-based applications, by preventing the activation of new, parallel threads from within transactions. Ricardo's current work is addressing this issue by developing an STM that supports any level of parallel nested transactions, while ensuring constant time overhead and retaining the consistency properties of the underlying STM.

4 Applications and Performance Evaluation

Thread mapping is an appealing approach to efficiently exploit the potential of modern chip-multiprocessors by taking advantage of their cores and memory hierarchy. However, these architectures are inherently complex and the efficiency of thread mapping relies upon matching the behavior of an application to system characteristics. In his talk, Márcio Castro proposed a dynamic mapping approach for TM applications which relies on a (off-line trained) machine learning algorithm [8] to determine, at runtime, if the current thread mapping strategy should be switched to a more adequate one. This dynamic approach was implemented within TinySTM and results were shown highlighting performance gains of up to 31% than the best static thread mapping for applications whose workload varies through different execution phases.

Diego Didona and Jörg Schenker discussed how to automatically identify the ”natural” degree of parallelism of an application [9], i.e., the threshold below which adding or removing threads will hamper performance. Their talk addressed both the scenario of centralized and distributed TM systems. In the centralized scenario, the concurrency degree is dynamically tuned using a gradient descent based on-line exploration approach. In distributed TM settings on-line exploration techniques incur in the prohibitive costs associated with state transfer. To circumvent this issue, they introduced a hybrid performance forecasting methodology, based on the sinergyc usage of machine learning and analytical modelling techniques [10].

Martin Schindewolf presented an experimental evaluation of TM on the BG/Q platform and the developments of new features in the CLOMP-TM benchmark, which included adding MPI routines for a hybrid OpenMP/MPI parallelization that exploits TM-based synchronization primitives. This talk also presented tools aimed at analyzing the performance of the TM run time system. These tools use the Blue Gene Performance Monitoring (BGPM) interface to correlate the statistics from the TM run time system with performance counter values, providing detailed insights in the run time behavior of TM applications.

João Soares talked about applying STM and software diversity in order to implement fault-tolerant components with minimal overhead. Faults are detected by concurrently executing operations in all replicas, and comparing the set of obtained results. An STM (Deuce using TL2) is used to preserve the causal order of these operations and guarantee that they all execute in the same order in all replicas.

The final talk of the workshop, presented by Maria Fazio, focused on the application of STM to simplify the development of resource brokers for IaaS Cloud platforms [12]. In these platforms, resource brokers are responsible for orchestrating the provisioning of resources based on user requests, which may be received and processed concurrently and having conflicting requirements. The resource brokers exploit Deuce STM to regulate concurrency among concurrent resource requests, which are specified according to XACML policies in order to offer a differentiated resource provisioning service. The proposed architecture has been developed in the context of the Cloud@Home project [4], which proposes a new volunteer computing paradigm, where the infrastructure is obtained by merging heterogeneous resources offered by different domains and/or providers.

5 Discussion: Where does TM research stand and what challenges lie ahead?

The workshop was concluded by a general discussion addressing a wide range of topics. These included controversial issues concerning the adoption of TM in complex applications, trade-offs in the design space of APIs for TM systems, as well as a retrospective analysis of recent developments in the areas of hardware, compilers and tool supports. Below we report some important points of these discussions (in chronological order):

5.1 Combining TM with blocking synchronization mechanisms

João Lourenço triggered a discussion on the need, in real-life scenarios, to have TM-based applications interact with external components, for instance via I/O or network. He pointed out that the lack of established solutions addressing this issue represents a major impairment to the adoption of TM.

Vincent Gramoli argued that it does not make sense to combine TM with other synchronization mechanisms (such as locks) as, achieving this result would imply exposing to the programmers much of the inherent complexity (e.g. reasoning on the possibility of deadlocks) that the TM abstraction is meant to hide.

Marc Shapiro suggested that there may be portions of an application for which TM represents the right abstraction and others in which a different paradigm (such as queues or event programming, for instance) fit better. So TM should be regarded as another tool in the toolbox and not as the ultimate solution for all applications.

5.2 STM going mainstream thanks to the recent GCC support?

Patrick Marlier, who has collaborated to define the specification of the recent GCC support for TM, was asked to highlight the key characteristics of the internal architecture of this tool.

Patrick explained that the GCC compiler generates code that will issue calls to the underlying TM algorithm. This allows decoupling GCC from a specific TM library, allowing the programmer to plug any TM library as long as it complies with the interfaces used by GCC. Of course GCC ships already with a default, lock-based TM library.

The development of this TM support at the GCC layer is expected to bring a host of benefits to the community. Due to the ubiquitous presence of GCC, new benchmarks and ideas for using TMs are very likely to arise. Besides, thanks to the open source nature of this software, the community can contribute to improve it by, for instance, implementing a richer API that supports more complex transaction types.

5.3 Which interfaces should STM expose to programmers?

The previous point led to the discussion on whether it would be beneficial for TM to expose richer, and therefore inherently more complex APIs, to allow programmers to maximize performance by triggering optimizations that could break potentially consistency in general scenarios.

Rachid Guerraoui commented on this and made a parallel with the development of object oriented programming languages. He argued that, in Smalltalk, the choice of granting excessing "freedom" to programmers led to the eventual decay of this programming language. JAVA, conversely, identified a sweet spot in the trade-off between complexity and features, which favoured its adoption in the software industry. He argued that it is an important challenge for this community to find the appropriate balance, designing interfaces that can be both safely usable by non-expert programmers and appealing to expert, performance craving programmers.

5.4 Hardware supports for TM: will they go mainstream... this time?

In 2009, the cancelling of the Rock processor, the first that planned to incorporate hardware support for TM, cast shadows on the adoption of this technology, at least in an imminent future. Today, these shadows seem to have been largely dispelled: IBM microprocessor Blue Gene/Q ships with hardware support for TM, and Intel has recently announced that its Haswell architecture, due to ship in 2013, will include hardware support for TM. This triggered a short discussion on how this compared with current research in the hardware TM field and on the impact this will have in the adoption of TM by a wider audience.

Regarding the first issue, it was pointed out that Intel's specifications, in their current form, are not sufficiently detailed to allow performing a fair comparison with state of the art solutions in the area.

On the second issue, there was unanimous consensus on that together with the release of GCC 4.7, these new processors will broaden significantly the audience of TM.

5.5 Benchmarks for TM: has this gap been filled?

Despite the number of efforts to develop complex benchmarks for STMs in recent times, João Lourenço pointed out that the spectrum of benchmarks available for TM systems is still very narrow. Most papers only present results using STAMP [6], micro-benchmarks (such as those provided by DSTM2 [18]), LeeTM [3] and STMBench7 [16].

Paolo Romano commented that this problem is particularly exacerbated in the area of distributed TM. Researchers working in this area are typically forced to perform non-trivial adaptations of benchmarks that were originally designed for centralized TM. This is not only a tedious but also a time consuming task. Benchmarks for centralized TMs, in fact, are not designed to scale to the high degrees of concurrency achievable in distributed settings. Further, it is arguable that the workloads generated by benchmarks originally designed for centralized TMs are actually representative of the ones that would be produced by applications developed from scratch to execute on a distributed TM platform.

5.6 Does TM really simplify the development of concurrent applications?

This question was addressed by Osman Unsal, who was involved in the development of QuakeTM [14] in the context of the Velox project [2], and presented his opinion on this topic. Osman reported that, at the time in which QuakeTM was being implemented, the development process took actually longer than if locks had been used. At that time, in fact, no tools (e.g. debuggers) were available to aid developers, and the only STM that would compile with QuakeTM had no source code. Hence, it was necessary to implement a host of tweaks to have a complex, real-time game such as Quake work on top of a TM stack.

However, compared to just a couple of years ago, nowadays many tools are freely available for the TM community, such as verification tools, transactional debuggers, programs that analyze transactional behavior, etc. Thanks to these latest developments, the development of TM applications has been drastically simplified.

References

[1] A. Adya. *Weak consistency: a generalized theory and optimistic implementations for distributed transactions*. PhD thesis, 1999. AAI0800775.

[2] Y. Afek, U. Drepper, P. Felber, C. Fetzer, V. Gramoli, M. Hohmuth, E. Riviere, P. Stenstrom, O. Unsal, W. M. Moreira, D. Harmanci, P. Marlier, S. Diestelhorst, M. Pohlack, A. Cristal, I. Hur, A. Dragojevic, R. Guerraoui, M. Kapalka, S. Tomic, G. Korland, N. Shavit, M. Nowack, and T. Riegel. The velox transactional memory stack. *IEEE Micro*, 30(5):76–87, Sept. 2010.

[3] M. Ansari, C. Kotselidis, K. Jarvis, M. Luján, C. Kirkham, and I. Watson. Lee-tm: A non-trivial benchmark for transactional memory. In *ICA3PP '08: Proceedings of the 7th International Conference on Algorithms and Architectures for Parallel Processing*. LNCS, Springer, June 2008.

[4] R. Aversa, M. Avvenuti, A. Cuomo, B. Di Martino, G. Di Modica, S. Distefano, A. Puliafito, M. Rak, O. Tomarchio, A. Vecchio, S. Venticinque, and U. Villano. The cloud@home project: towards a new enhanced computing paradigm. In *Proceedings of the 2010 conference on Parallel processing*, Euro-Par 2010, pages 555–562, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[6] C. Cao Minh, J. Chung, C. Kozyrakis, and K. Olukotun. STAMP: Stanford transactional applications for multi-processing. In *IISWC '08: Proceedings of The IEEE International Symposium on Workload Characterization*, September 2008.

[7] F. M. Carvalho and J. Cachopo. Adaptive object metadata to reduce the overheads of a multi-versioning stm. In *Multiprog-2012*, Paris, january 2012.

[8] M. Castro, L. F. W. Góes, C. P. Ribeiro, M. Cole, M. Cintra, and J.-F. Méhaut. A Machine Learning-Based Approach for Thread Mapping on Transactional Memory Applications. In *High Performance Computing Conference (HiPC)*, pages 1–10, Bangalore, India, 2011. IEEE Computer Society.

[9] D. Didona, P. Felber, D. Harmanci, P. Romano, and J. Schenker. Elastic scaling for transactional memory: From centralized to distributed architectures (poster). In *4th USENIC Workshop on Hot Topics in Parallelism (Hotpar'12)*, Berkeley, Ca, USA, June 2012.

[10] D. Didona, P. Romano, S. Peluso, and F. Quaglia. Transactional auto scaler: Elastic scaling of nosql transactional data grids. Technical Report 50, INESC-ID, December 2011.

[11] N. Diegues and J. a. Cachopo. Parallel nesting in a lock-free multi-version software transactional memory. In the 7th ACM SIGPLAN Workshop on Transactional Computing (TRANSACT), 2012.

[12] M. Fazio and A. Puliafito. Virtual resource management based on software transactional memory. In *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, pages 1 –8, nov. 2011.

[13] S. M. Fernandes and J. Cachopo. Lock-free and scalable multi-version software transactional memory. In *16th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 179–188. ACM SIGPLAN, february 2011.

[14] V. Gajinov, F. Zyulkyarov, O. S. Unsal, A. Cristal, E. Ayguade, T. Harris, and M. Valero. Quaketm: parallelizing a complex sequential application using transactional memory. In *Proceedings of the 23rd international conference on Supercomputing*, ICS '09, pages 126–135, New York, NY, USA, 2009. ACM.

[15] V. Gramoli and R. Guerraoui. Brief announcement: transaction polymorphism. In *SPAA*, pages 311–312, 2011.

[16] R. Guerraoui, M. Kapalka, and J. Vitek. STMBench7: a benchmark for software transactional memory. *SIGOPS Operating Systems Review*, 41(3):315–324, 2007.

[17] R. Guerraoui and A. Schiper. Genuine atomic multicast in asynchronous distributed systems. *Theoretical Computer Science (Elsevier)*, 254:297–316, 2001.

[18] M. Herlihy, V. Luchangco, and M. Moir. A flexible framework for implementing software transactional memory. In *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, OOPSLA '06, pages 253–262, New York, NY, USA, 2006. ACM.

[19] G. Korland, N. Shavit, and P. Felber. Deuce: Noninvasive software transactional memory in java. *Transactions on HiPEAC*, 5(2), 2010.

[20] W. Maldonado, P. Marlier, P. Felber, J. L. Lawall, G. Muller, and E. Riviere. Deadline-aware scheduling for software transactional memory. In *DSN*, pages 257–268, 2011.

[21] R. Palmieri, F. Quaglia, and P. Romano. Osare: Opportunistic speculation in actively replicated transactional systems. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*, pages 59 –64, oct. 2011.

[22] R. Palmieri, F. Quaglia, P. Romano, and N. Carvalho. Evaluating database-oriented replication schemes in software transactional memory systems. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1 –8, april 2010.

[23] S. Peluso, P. Ruivo, P. Romano, F. Quaglia, and L. Rodrigues. When scalability meets consistency: Genuine multiversion update serializable partial data replication. In *32nd International Conference on Distributed Computing Systems (ICDCS'12)*, Macao, China, June 2012.

[24] H. Rito and J. P. Cachopo. Memoization of methods using software transactional memory to track internal state dependencies. In *PPPJ*, pages 89–98, 2010.

[25] P. Romano, N. Carvalho, and L. Rodrigues. Towards distributed software transactional memory systems. In *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, LADIS '08, pages 4:1–4:4, New York, NY, USA, 2008. ACM.

[26] C. Seaton, D. Goodman, M. Luján, and I. Watson. Applying dataflow and transactions to Lee routing. In *Workshop on Programmability Issues for Heterogeneous Multicores*, 2012.

[27] J.-T. Wamhoff, C. Fetzer, P. Felber, E. Rivière, and G. Muller. Fastlane: Streamlining transactions for low thread counts. In *TRANSACT 12*, New Orleans, LA, USA, June 2012. ACM New York, NY, USA.

[28] P. T. Wojciechowski. Extending atomic tasks to distributed atomic tasks. In *Proceedings of* $(EC)^2$ *'08: Workshop on Exploiting Concurrency Efficiently and Correctly (co-located with CAV '08: the 20th Int. Conf. on Computer Aided Verification, Princeton, USA)*, July 2008.