Scientific Report for STSM project

Alexander Matveev
matveeva@post.tau.ac.il

December 11, 2014

COST Action	IC1001
STSM title	Hybrid Transactional Memory for Haswell
Reference	COST-STSM-ECOST-STSM-IC1001-130914-047641
STSM dates	From 13/09/2014 to 01/11/2014
Location	Neuchâtel, Switzerland
Participant	Alexander Matveev
Home institution	Tel-Aviv University
Host institution	University of Neuchâtel
Host supervisor	Prof. Pascal Felber

1 Context

Intel have recently released hardware support for best-effort hardware transactional memory (HTM) in their latest Haswell processors [10]. Best-effort HTMs impose limits on hardware transactions, but eliminate the overheads associated with loads and stores in software transactional memory (STM) implementations. Because it is possible for HTM transactions to fail for various reasons, a hybrid transactional memory (HyTM) approach has been studied extensively in the literature. It supports a best effort attempt to execute transactions in hardware, yet always falls back to slower all-software transactions in order to provide better progress guarantees and the ability to execute various systems calls and protected instructions that are not allowed in hardware transactions.

The first HyTM [4, 5] algorithms supported concurrent execution of hardware and software transactions by instrumenting the hardware transactions' shared reads and writes to check for changes in the STM's metadata. This approach, which is the basis of all the generally applicable HyTM proposals, imposes severe overheads on the hardware transaction—the HyTM's frequently executed *fast-path*.

Riegel et al. [8] provide an excellent survey of HyTM algorithms to date, and the various proposals on how to reduce the instrumentation overheads in the hardware fast-path. There are three key proposed approaches, each with its own limitations.

The first is Phased TM [6], in which transactions are executed in phases, each of which is either all hardware or all software. Phased TM performs well when all hardware transactions are successful, but has poor performance if even a single transaction needs to be executed in software, because it must switch all transactions to a slower all-software mode of execution. Though this is a good approach for some workloads, in general it is not clear how to overcome frequent switches between phases.

The second approach, Hybrid Norec [2], is a hybrid version of the efficient Norec STM [3]. In it, write transactions' commits are executed sequentially and a global clock is used to notify concurrent read transactions about the updates to memory. The write commits trigger the necessary re-validations and aborts of the concurrently executing transactions. The great benefit the Norec HyTM scheme over classic HyTM proposals is that no metadata per memory location is required and instrumentation costs are reduced significantly. However, as with the original Norec STM, scalability is limited

because the conflicts cannot be detected at a sufficiently low granularity.

The third approach, by Riegel et al. [8], effectively reduces the instrumentation overhead of hardware transactions in HyTM algorithms based on both the LSA [9] and Norec [3] STMs. It does so by using non-speculative operations inside the hardware transactions. Unfortunately, these operations are supported by AMD's proposed ASF transactional hardware [1] but are not supported in the best-effort HTM that Intel is bringing to the marketplace.

Recently, a new *reduced hardware* (RH) transactions approach have been proposed by [7]. RH transactions allow an extensive reduction of the instrumentation overhead of the hardware fast-path transactions, without impairing concurrency and safety among hardware and software transactions.

1.1 STSM motivation and goal

Intel Haswell and IBM Power8 provide support for atomic hardware transactions, and our goal is to use this hardware atomicity to simplify and reduce the required coordination in the software.

The host institute in Neuchâtel (UniNE) has strong expertise in HyTM, notably because they were part of the team that designed ASF-based HyTM algorithms [8] in the context of the European FP7 project VELOX. The applicant is the main developer of the newly proposed reduced hardware TM design [7]. Therefore, combining the skills of both groups is likely to result in significant advances in designing new HyTM algorithms that are both correct and efficient on current multicore processors.

Preliminary experiments conducted by Patrick Marlier (post-doc in Prof. Felber's group) indicate that it is difficult to provide correctness and efficiency for HyTM systems. New problems arise as a result of hardware capacity limitations, system hardware events, unsupported instructions, and so on.

Our goal in this STSM is to construct a new correct and efficient HyTM algorithm that can work on current HTM systems for as many applications as possible. In particular, we will investigate different tradeoffs: HyNOrec vs. HyTL2/HyLSA, reduced-hardware vs. full-software, eager TM vs. lazy TM, various retry policies, and lock-based vs obstruction-free designs.

2 Description of the work carried out

In the first 2 weeks, we designed a new Hybrid TM algorithm that uses the RH approach to provide low-overhead coordination between the hardware fast-path transactions and the software slow-path fallbacks. The core idea is (1) to use multiple small hardware transactions in the software slow-path, and (2) to provide a fully pessimistic slow-path: each slow-path transaction executes only once and never aborts. This new fully pessimistic hybrid mechanism provides the ability to replace lock-based critical sections in existing applications, without the need to perform complex code rewrites, like it is required by the standard Hybrid TM systems.

In the next 4 weeks, we implemented the new hybrid algorithm as an external library, and then applied it to a number of existing state-of-the-art concurrent implementations: the RCU-based linked-list, hash-table and Citrus tree. These RCU-based data-structures are widely used in the Linux Kernel to provide scalability for multicore machines. So, if the new hybrid system can reach or even win over the RCU scheme, then it provides a better alternative to the current state-of-the-art.

In Figure 1, we present throughput results on the new 16-way Intel HTM Haswell processor for the RCU-based and RWLock-based (reader-writer lock) linked-list and hash-table, and the RCU-based Citrus tree. We compare these results to the new hybrid implementations: the RH-HyTM and HTM-RH-HyTM. The first hybrid is the basic full-software implementation that does not use any hardware transactions. However, the second uses hardware transactions and provides much better results. The 10, 20 and 40 graph variants correspond to 10%, 20% and 40% mutation ratios: the relative amount of write transactions. We can see that the HTM performance boost is really significant, and the new hybrid is 1.5 and 2.0 times faster than the RCU based linked-list and hash-table. For the Citrus tree, the hybrid is close to the original performance. However, the original Citrus tree is highly optimized fine-grained lock-based implementation, while the hybrid implementation has no manual locking at all. So, the new hybrid provides close to optimal results without the need to use tedious locking techniques.

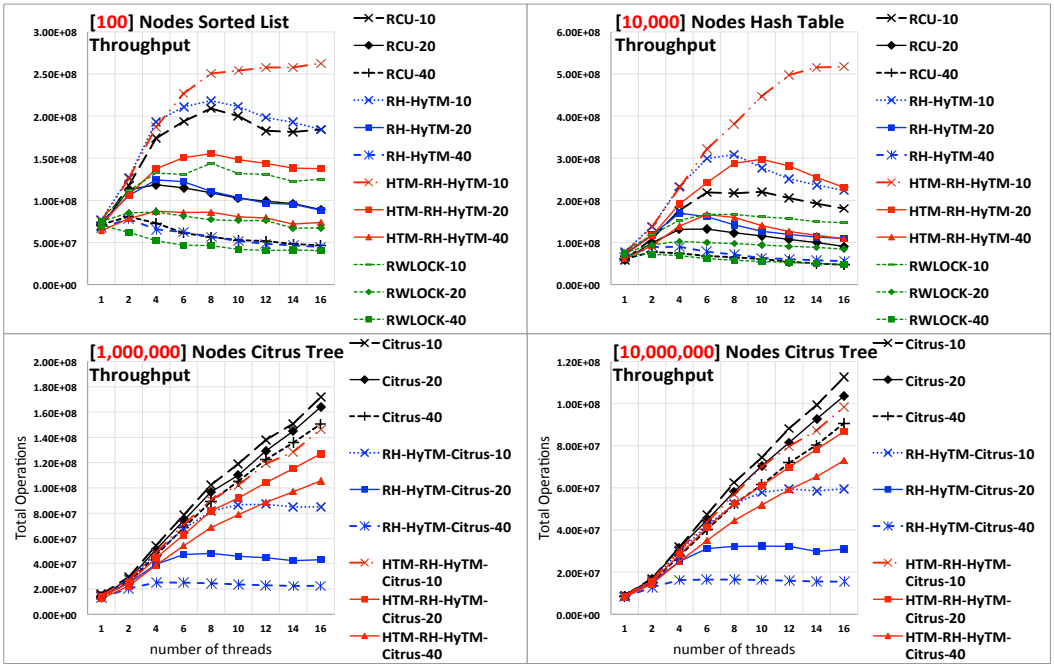

Figure 1: Throughput results for the linked-list, hash-table and Citrus-Tree. The X-axis represents the number of threads, and the Y-axis represents the total amount of operations completed.

3 Future collaboration

Our next step is to apply the new hybrid scheme to popular user-space applications. In particular, we currently integrate the new scheme into the Kyoto Cache DB and the Memcached applications, that are widely used by popular websites, like Facebook and Youtube.

Then, we plan to automate the application of this system by using the GCC TM automatic instrumentation. In this way, the programmer can apply the hybrid by enclosing the code block into the *__transaction_atomic* { .. } statement, which is similar to the simple but not scalable one big critical section.

Finally, we are going to port and optimize our new hybrid system to the new 96-way IBM Power8 HTM system, that provides more flexibility than the Intel Haswell 16-way CPU.

4 Confirmation by the host institution of the successful execution of the STSM

The host Prof. Pascal Felber from UniNE confirms that Alexander Matveev achieved all the target objectives that we defined for this collaboration with distinction. He analyzed and identified performance bottlenecks, and proposed novel Hybrid TM algorithms. He developed a new, very promising strategy that is still being refined for an upcoming publication. The host will send further confirmation on the successful execution of the STSM directly to Euro-TM Chair.

References

[1] D. Christie, J.-W. Chung, S. Diestelhorst, M. Hohmuth, M. Pohlack, C. Fetzer, M. Nowack, T. Riegel, P. Felber, P. Marlier, and E. Rivière. Evaluation of amd's advanced synchronization facility within a complete transactional memory stack. In *Proceedings of the 5th European conference on Computer systems*, pages 27–40, New York, NY, USA, 2010. ACM.

[2] L. Dalessandro, F. Carouge, S. White, Y. Lev, M. Moir, M. L. Scott, and M. F. Spear. Hybrid norec: a case study in the effectiveness of best effort hardware transactional memory. *SIGPLAN Not.*, 46(3):39–52, Mar. 2011.

[3] L. Dalessandro, M. F. Spear, and M. L. Scott. Norec: streamlining stm by abolishing ownership records. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '10, pages 67–78, New York, NY, USA, 2010. ACM.

[4] P. Damron, A. Fedorova, Y. Lev, V. Luchangco, M. Moir, and D. Nussbaum. Hybrid transactional memory. *SIGPLAN Not.*, 41(11):336–346, Oct. 2006.

[5] S. Kumar, M. Chu, C. J. Hughes, P. Kundu, and A. Nguyen. Hybrid transactional memory. In *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '06, pages 209–220, New York, NY, USA, 2006. ACM.

[6] Y. Lev, M. Moir, and D. Nussbaum. Phtm: Phased transactional memory. In *In Workshop on Transactional Computing (Transact), 2007. research.sun.com/scalable/pubs/ TRANSACT2007PhTM.pdf*, 2007.

[7] A. Matveev and N. Shavit. Reduced hardware transactions: a new approach to hybrid transactional memory. In *SPAA 2013*, 2013.

[8] T. Riegel, P. Marlier, M. Nowack, P. Felber, and C. Fetzer. Optimizing hybrid transactional memory: the importance of nonspeculative operations. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, SPAA '11, pages 53–64, New York, NY, USA, 2011. ACM.

[9] P. F. T. Riegel and C. Fetzer. A lazy snapshot algorithm with eager validation. In *20th International Symposium on Distributed Computing (DISC)*, September 2006.

[10] Web. Intel tsx http://software.intel.com/en-us/blogs/2012/02/07/transactional-synchronization-in-haswell, 2012.