The moment of truth: are we done with STM?

Nuno Diegues
INESC-ID / IST, Lisbon, Portugal
ndiegues@gsd.inesc-id.pt

Paolo Romano
INESC-ID / IST, Lisbon, Portugal
romano@inesc-id.pt

1 Commodity HTM - Intel TSX

The advent of Transactional Memory has reached a significant milestone with the release of Intel's Haswell processors, which contain support for Hardware Transactional Memory (HTM) through TSX [5]. After over a decade of intense research, which resulted in numerous advances in Software TM implementations (STMs), we are now faced with a question: now that HTM has become mainstream, what role will STM come to play in the future? Will the advent of HTM determine the end of STM (and of research in the area of STM)?

We seek to answer this question by conducting a large experimental study on synchronization with locks and TM variants. Now that TM is widely accessible to programmers, our goal is to establish the current strengths and flaws of the available TM mechanisms, and thus to better understand where we are in the path of making concurrent programming an easier task. For this, we explore a wide variety of synchronization mechanisms, at different levels of abstraction, and assess their efficiency from the twofold perspective of performance and power consumption:

• Locks — Decades of synchronization with locks have conveyed many alternative implementations, many times trading off subtle changes with great impact in performance. Hence we seek to consider both coarse and fine-grained locking (not shown here for space constraints) approaches, by relying on 6 different lock implementations [3].

• STM — After a decade of intense and successful advances in research, many software implementations have consolidated the idea of TM. We chose four such STMs, with different characteristics, that are widely accepted as state of the art (TL2, NOrec, TinySTM and SwissTM).

• HTM — The release of TSX in the latest Intel processor raises the expectations of a wide dissemination of hardware support for TM. Therefore we considered that HTM (there exist alternatives by IBM, although they are not as accessible as Intel's commodity hardware). TSX is a best-effort HTM, which means that one cannot rely only on the HTM, as a transaction may never succeed. For this reason, we followed the approach proposed by Intel of using locks as a fall-back strategy when the hardware is not successful. We use both a single lock or fine-grained locks (which require *per-application* effort).

• Hybrid TM (HyTM) — Another mechanism proposed in the literature is to use HTM and STM in synergy. The rationale is once again that HTM has best-effort semantics, and thus using an STM on the fall-back can ensure progress. In this case we considered the two state of the art Hybrid TM proposals [4, 2] which, coincidently, rely on two of the STMs that we also considered: TL2 and NOrec.

2 Preliminary Study

We used the suite of transactional benchmarks STAMP, which is known for having a wide variety of applications, ranging from low to high contention, small to large transactions[1]. The machine used in the study is equipped with a Intel Haswell Xeon E3-1275 3.5Ghz processor with 32GB RAM, which includes 8 virtual cores (4 physical, with hyper-threading) and the recent TSX HTM support. This is the maximum degree of parallelism currently

[1] For the time being, we had to exclude Kmeans from the study, as its TSX-based porting suffers of a bug that is still being investigated.

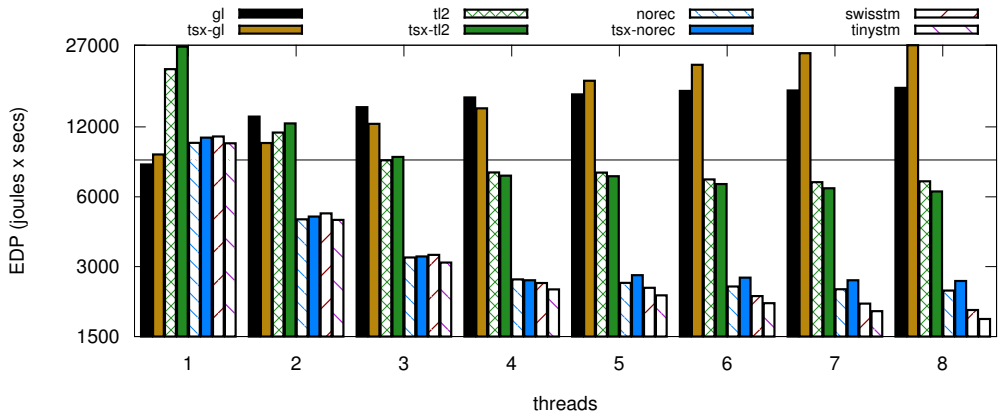

Figure 1: Average EDP across all STAMP benchmarks (but KMeans). The horizontal line shows the EDP of a non-instrumented single threaded execution.

achievable with Intel processors supporting TSX, and, although it is expectable that this limitation will be relaxed in the near future, this experimental platform still allows to gain interesting insights on the existing trade-offs among the considered synchronization mechanisms. In our study we consider the Energy Delay Product (EDP) metric, which combines both performance and power consumed. Fig. 1 shows the EDP of each solution, and a constant line with the EDP of a non-instrumented execution with 1 thread (used as baseline). Note that TSX approaches were highly optimized with the following recently published contributions [4, 5, 1].

Note that no mechanism achieves the same efficiency as a non-instrumented execution, except for a single global lock (GL) and TSX with that fallback (TSX-GL). However, as concurrency grows, all approaches steadily reduce their EDP except for GL and TSX-GL. Among these, TinySTM is always on the lead for reducing EDP. SwissTM, NOrec and TSX+NOrec are very close to TinySTM until 4 threads. Beyond that point the TSX based approach lags behind due to the contention on hyper-threads in the same L1 cache. Also, TL2-based approaches perform significantly worse (even comparing to TSX-GL).

If we consider each STAMP application on its own instead (not shown for space constraints), then there is no clear winner. Except for a single lock, all other classes of mechanisms are the best in at least one benchmark: STM in Genome, Labyrinth, Vacation and Yada; HTM in SSCA2; HyTMs in Intruder. At this point these results clearly show that STM is the best solution with higher concurrency, and that STM is can be quite competitive even at a low thread count. Clearly, these results are preliminary, and we are eager to be able to evaluate the efficiency of TSX on architectures supporting higher degrees of parallelism. Nonetheless, this experimental study highlights that, at least with the currently available HTM supports, STM will still play an important role.

References

[1] Yehuda Afek, Amir Levy, and Adam Morrison. Programming with hardware lock elision. In *Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '13, pages 295–296, 2013.

[2] 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 XVI, pages 39–52, 2011.

[3] Tudor David, Rachid Guerraoui, and Vasileios Trigonakis. Everything you always wanted to know about synchronization but were afraid to ask. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 33–48, 2013.

[4] Alexander Matveev and Nir Shavit. Reduced hardware transactions: a new approach to hybrid transactional memory. In *Proceedings of the 25th ACM symposium on Parallelism in algorithms and architectures*, SPAA '13, pages 11–22, 2013.

[5] Richard Yoo, Christopher Hughes, Konrad Lai, and Ravi Rajwar. Performance evaluation of intel transactional synchronization extensions for high-performance computing. In *Proceedings of the 2013 International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '13, 2013.