Scientific Report for STSM project

Diego Didona
didona@gsd.inesc-id.pt

December 4, 2014

COST Action	IC1001
STSM title	Black-box self-tuning of TM applications
Reference	ECOST-STSM-IC1001-031114-051580
STSM dates	From 03-11-2014 to 10-11-2014
Location	École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Participant	Diego Didona
Home institution	Instituto Superior Técnico, Lisbon, Portugal
Host institution	École Polytechnique Fédérale de Lausanne
Host supervisor	Prof. Rachid Guerraoui

1 Context

The advent of multi-core architectures has brought parallel computing to the fore-front of software development, fostering intense research on programming paradigms aimed at simplifying the development of concurrent applications. Transactional Memory (TM) is arguably the most prominent proposal in this sense. The growth of popularity and relevance of the TM paradigm have recently led to the standardization of constructs for TM in popular program- ming languages (such as C and C++), as well as to the integration of TM supports in mainstream processors by Intel [11] and IBM [8].

Performance of TM, however, is a more controversial matter. In fact, the literature on TM has shown, on the one hand, that TM can compete with complex locking strategies [5], and, even outperform them in irregular applications [3] for which it is hard to conceive effective fine-grained locking protocols. On the other hand, the vastness of the design space of TM has led to the exploration of a large number of algorithmic alternatives, as well as to implementations based purely on software (STM), hardware (HTM), and combinations thereof. Indeed, the efficiency of any existing TM implementation is strongly dependent on the characteristics of the workloads generated by TM applications, and can vary due to sensitivity to program inputs [5],or phases of program execution [3, 6]. The efficiency of TM implementations is also influenced by the correct tuning of a number of internal parameters of TM algorithms (e.g., number of active threads or of retries using HTM), as well as by architectural aspects of the underlying hardware (e.g., single-chip vs multi-chip systems). Overall, the search for a one-size-fits-all-solution, capable of ensuring optimal performance across all possible TM workloads, has, so far, been unsuccessful. Also, as heterogeneity of hardware platforms increases, the possibility of identifying any single TM algorithm as "best" grows increasingly unlikely [10].

In the light of the aforementioned considerations, many research efforts have been devoted in the last years to devise solutions for the self tuning of TM applications. The following section focuses on briefly surveying existing solutions in the field.

1.1 Self-tuning in TM

Proposed techniques in the area of self-tuning TM applications rely on different methodologies and target different parameters of the TM design space.

One of the most investigated problem is determining the optimal number of active threads. Proposed solutions rely on Analytical Modeling [1], off-line Machine Learning [9], or on exploration-based strategies [3].

Wang et al. [10] propose to exploit Artificial Neural Networks to determine the best TM implementation depending on the workload. Their solution relies on a preliminary analysis of the target application's code to extract a set of *static* features (e.g., number of transactional blocks) and on the on-line profiling of the application to measure a set of *dynamic* features (e.g., percentage of read-only transactions). Such dynamic workload characterization phase relies on the availability of an ad-hoc TM implementation, which is exploited only during the on-line profiling phase.

Felber et al. [7] focus on the effect of lock granularity on performance, where lock granularity refers to the amount of data that is guarded by a single lock, and apply a hill climbing algorithm to implement the self-tuning of such parameter.

Tuner [6] exploits a Reinforcement Learning algorithm in order to adaptively determine the optimal retry-on-abort policy to adopt for Intel HTM-based applications.

1.2 STSM motivation and goal

The conducted survey on literature about TM optimization has highlighted that, while very heterogeneous in the employed techniques, all the proposed approaches share the main limitation of targeting the self-tuning of only one parameter of a TM; this leads those approaches to deliver, ultimately, sub-optimal performance. A natural research challenge that stems from this observation is devising an autonomic scheme that is able to optimize a TM as a whole, self-tuning many of its internal parameters.

Another important conclusion that has been drawn after conducting the study of the literature on TM self-tuning is that existing solutions can be divided in two categories: model-free and model-based.

Model-free solutions try to optimize a TM by means of exploration, i.e., by trying different configurations and selecting, in the end, the one that maximizes performance. The advantage of this scheme is that is very lightweight, as it only needs to measure the performance corresponding to a given configuration, and the rule to apply to determine the next configuration to explore or whether it is the case to stop exploration is incrementally defined in an on-line fashion, i.e., without any *a priori* knowledge on the target applica-

tion. On the downside, model-free techniques are cumbersome to apply in optimization problems spanning multiple dimensions, as the transitory phase needed to carry out exploration would grow with the total number of possible configurations.

Model-based solutions, conversely, require an extensive calibration phase before being deployed with the application: such phase can coincide with the derivation of an analytical model of the target system/application or a training phase in the case of Machine Learning-based solutions. Moreover, in order to be instantiated at runtime, the models need a detailed workload characterization, which is achieved by means of code instrumentation that inevitably results in additional overhead [10]. On the other hand, once instantiated, a self-tuning scheme based on an accurate performance model may optimize an application needing no or a very little amount of explorations.

Model-free and model-based approaches are, thus, seen as antithetic and, to the best of our knowledge, no solutions reconciling the two has ever been proposed in the field of self-tuning mechanism for TM applications.

The long-term goal of the collaboration between the Home and the Host institution is to investigate black box self-tuning techniques that achieve the two-fold goal of *i*) optimizing several parameters of the TM middleware, and not just one like existing solution; *ii*) combining the model-based and the model-free approach, so as to achieve the best of the two worlds, i.e., the predictive power of the first methodology and the lightweight and prompt instantiation of the second one.

Given its short duration –only one week– this STSM has mainly served as bootstrapping phase to foster cross-fertilization and knowledge transfer between the Host's and the Home's research groups.

2 Description of the work carried out

Three are the activities carried out during this STSM.

1. In the context of my PhD, I have been investigating the problem of self-tuning distributed TMs, applying techniques that rely either on the synergistic use of analytical modeling and machine learning (e.g., [4]), or on purely exploration-based approaches (e.g., [2]). On the other hand, Prof. Guerraoui is a renowned researcher in the field of centralized TM, with strong background in both theoretical and

implementation-related aspects of this programming model. The first phase of the STSM has been dedicated to reason about the applicability of techniques that I have already implemented to the problem of self-tuning centralized TM.

2. As briefly introduced, the long-term research plan envisioned by the Host and the Home institution encompasses devising a self-tuning scheme that is able to optimize different parameters of a TM, including the number of active threads and the TM implementation itself. The Home institution has available some machines equipped with a Haswell architecture, i.e., with support for Intel Hardware TM; however, such machines are only equipped with 8 cores, thus limiting the amount of available parallelism. The Host institution, instead, has a 48 cores machine at disposal, thus enabling the execution of (and consequently allowing for the the self-tuning of) highly scalable applications. The second task of the STSM consisted, thus, into setting up and adapting the TM benchmarking framework developed at the Home institution [5] to the Host institution's machine.

3. The third task of the STSM consisted into enhancing the statistics monitoring and collection module of the TM implementations included in the benchmarking framework. In particular, we have implemented a superset of the workload features (referred to as *dynamic features* in the original paper) exploited by the solution proposed by Wang et al. [10]. This will give us the opportunity to investigate workload-characterization based techniques and to compare on equal terms with a state-of-the-art solution for TM optimization [10]. In carrying out this task I greatly benefit from the guidance and support of the members of Prof. Guerraoui's research team. In fact, in carrying out my research, I have always worked on Java implementations of distributed TMs, thus not being very acquainted with the subtle details of C and C++ implementations of centralized TMs like NORec, TinySTM, SwissTM and TL2. On the other side, Prof. Guerraoui's lab members proved to be expert in the art of hacking TM implementations.

3 Future collaboration

The Home and the Host institutions are still working in synergy to explore the research lines outlined in the previous section, with the final goal of publishing a paper describing the achieved results.

References

[1] P. Di Sanzo et al. Regulating concurrency in software transactional memory: An effective model-based approach. In *Proc. of SASO*, 2013.

[2] D. Didona et al. An extremum seeking algorithm for message batching in total order protocols. In *In Proc. of SASO*, 2012.

[3] D. Didona et al. Identifying the optimal level of parallelism in transactional memory applications. *Springer Computing Journal*, 2013.

[4] D. Didona et al. Transactional auto scaler: Elastic scaling of replicated in-memory transactional data grids. *ACM Trans. Auton. Adapt. Syst.*, 9(2):11:1–11:32, July 2014.

[5] N. Diegues et al. Virtues and limitations of commodity hardware transactional memory. In *Proc. of PACT*, 2014.

[6] N. Diegues and P. Romano. Self-tuning intel transactional synchronization extensions. In *Proc. of ICAC*, 2014.

[7] P. Felber et al. Dynamic performance tuning of word-based software transactional memory. In *Proc. of PPoPP*, 2008.

[8] C. Jacobi, T. Slegel, and D. Greiner. Transactional memory architecture and implementation for ibm system z. In *Proceedings of the Annual nternational Symposium on Microarchitecture (MICRO)*, pages 25–36. IEEE Computer Society, 2012.

[9] D. Rughetti et al. Machine learning-based self-adjusting concurrency in software transactional memory systems. In *Proc. of MASCOTS*, 2012.

[10] Q. Wang et al. A transactional memory with automatic performance tuning. *ACM TACO*, 8(4), Jan. 2012.

[11] R. M. Yoo, C. J. Hughes, K. Lai, and R. Rajwar. Performance evaluation of intel® transactional synchronization extensions for high-performance computing. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–19. ACM, 2013.