Adaptive Transactional Memories: Performance and Energy Consumption Trade-offs

Diego Rughetti, Pierangelo Di Sanzo, Alessandro Pellegrini

High Performance and Dependable Computing Systems Research Group

Sapienza Università di Roma

The Study

Energy efficiency is a pressing issue, especially in large data centers:

- non-negligible management cost
- enhancement of hardware fault probability
- significant environmental footprint

Can Software Transactional Memory (STM) provide benefits on both power saving and the overall applications' execution performance?

Encapsulating shared-data accesses within transactions gives the freedom to the STM middleware to both **ensure consistency** and **reduce the actual data contention**, the latter having been shown to affect the **overall power needed to complete the application's execution**.

Reference Implementations

- TinySTM [1] — implements the Encounter-Time Locking (ETL) algorithm. It relies on a shared array of locks, where each lock is associated with a portion of the (shared) address space.
  - SAC-STM [2] — exploits a machine-learning based controller which regulates the amount of active concurrent threads along the execution of the application. A neural network is trained to learn relations between the average wasted transaction execution time, a set of workload-profile parameters, and the number of active concurrent threads.
  - SCR-STM [3] — similar in spirit to SAC-STM, yet it relies on an analytic model to let the controller regulate the concurrency level.
  - ATS-STM [4] — a transaction-scheduling algorithm relying on runtime measurement of the Contention Intensity (CI), which is re-calculated whenever a transaction commits or aborts. Before starting a new transaction, if the current value of CI exceeds a given threshold, then the thread stalls and the transaction is inserted within a queue shared by all threads.
  - Shrink [5] — a transaction-scheduling algorithm, based on temporal locality. If the transaction success rate is below a given threshold, contention probability is evaluated on the read- and write-sets, which determines whether the new transaction must be serialized.
- R-STM [6] — is a transactional memory middleware which allows adaptivity on two different sides. On the one hand, it implements a coarse-grained adaptivity system, that allows to change the STM implementation being used during the execution of the application. On the other hand, once a particular STM implementation is selected, it allows to fine tune the execution parameters for an active transaction.

Benchmark Applications

- intruder — implements a signature-based network intrusion detection systems (NIDS). Three analysis phases are carried on in parallel: *capture*, *reassembly*, and *detection*. Only the first two are enclosed by transactions. Overall, the total amount of time spent in the execution of transactions is relatively moderate.
- yada — implements Ruppert's algorithm for Delaunay mesh refinement. This benchmark shows a high level of parallelism, and transactional operations involve only updates of the shared mesh representation and cavity expansion. The overall execution time of this benchmark is relatively long, with long transactions and a significantly higher number of memory operations.

Learning/Scheduling Times

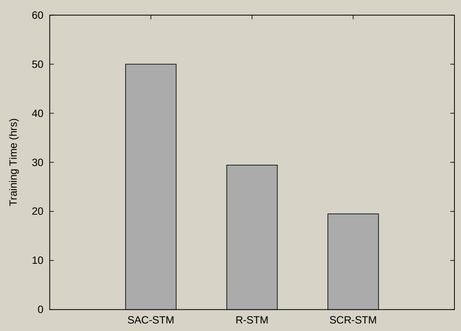

Figure: Training Time, intruder

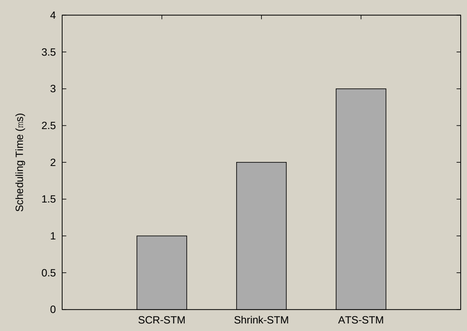

Figure: Scheduling Time, intruder

References

[1] Pascal Felber, Christof Fetzer, and Torvald Riegel.
Dynamic performance tuning of word-based software transactional memory.
In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP, pages 237–246. ACM, 2008.

[2] Diego Rughetti, Pierangelo Di Sanzo, Bruno Ciciani, and Francesco Quaglia.
Machine learning-based self-adjusting concurrency in software transactional memory systems.
In *Proceedings of the 20th IEEE International Symposium On Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS, pages 278–285. IEEE Computer Society, August 2012.

[3] Pierangelo Di Sanzo, Francesco Del Re, Diego Rughetti, Bruno Ciciani, and Francesco Quaglia.
Regulating concurrency in software transactional memory: An effective model-based approach.
In *Proceedings of the 7th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, SASO. IEEE Computer Society, September 2013.

[4] Richard M. Yoo and Hsien-Hsin S. Lee.
Adaptive transaction scheduling for transactional memory systems.
In *Proceedings of the 20th Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA, pages 169–178. ACM, 2008.

[5] Aleksandar Dragojević, Rachid Guerraoui, Anmol V. Singh, and Vasu Singh.
Preventing versus curing: Avoiding conflicts in transactional memories.
In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, PODC, pages 7–16. ACM, 2009.

[6] Michael F. Spear.
Lightweight, robust adaptivity for software transactional memory.
In *Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA, pages 273–283. ACM, 2010.

Experimental results for the intruder benchmark

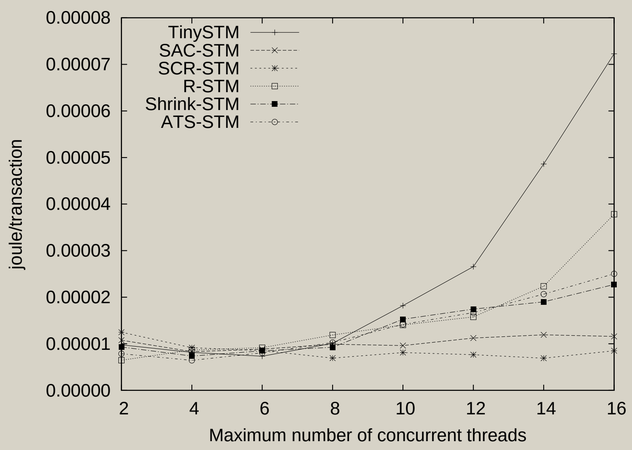

Figure: Per-Transaction Energy Consumption

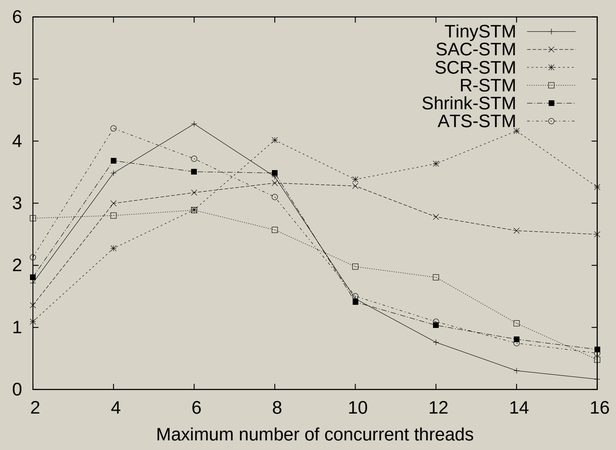

Figure: Throughput/Energy Consumption Ratio

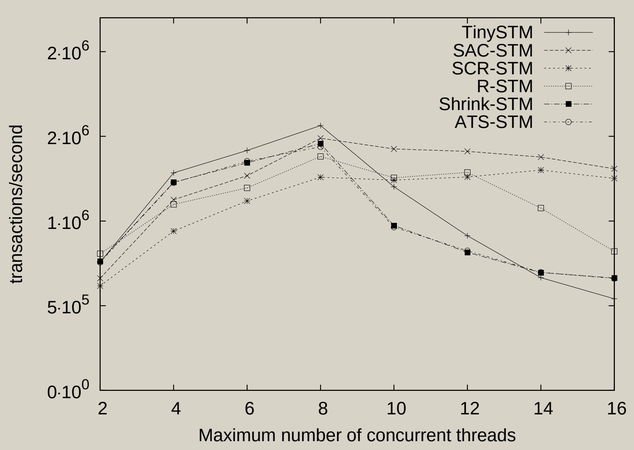

Figure: Application Throughput

Experimental results for the yada benchmark

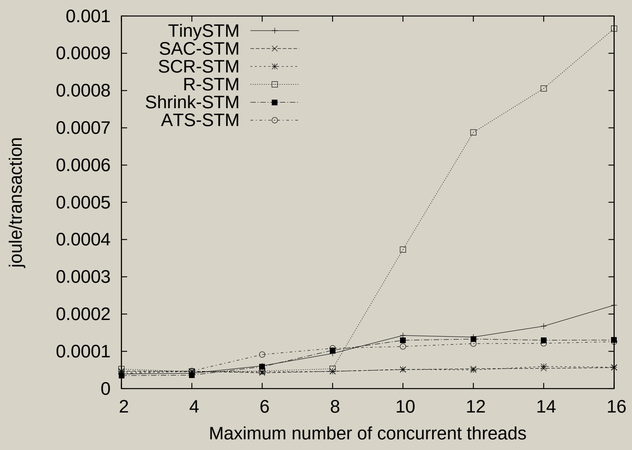

Figure: Per-Transaction Energy Consumption

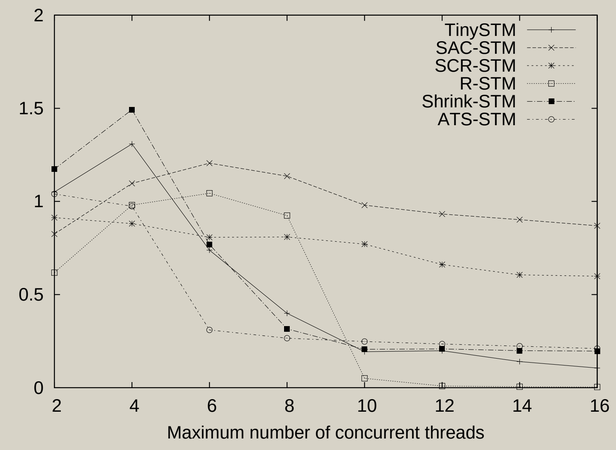

Figure: Throughput/Energy Consumption Ratio

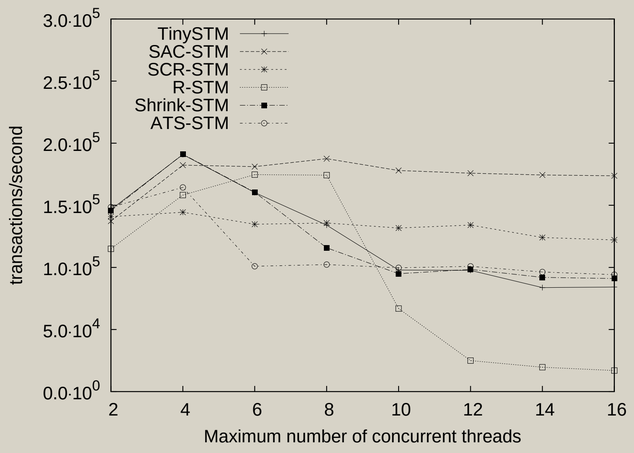

Figure: Application Throughput

Adaptivity is a strictly necessary requirement to reduce energy consumption in STM systems

http://www.dis.uniroma1.it/∼pellegrini pellegrini@dis.uniroma1.it