Adaptive Transactional Memories: Performance and Energy Consumption Trade-offs

Diego Rughetti, Pierangelo Di Sanzo, Alessandro Pellegrini
High Performance and Dependable Computing Systems Research Group

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 into 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] — a transactional memory middleware which allows adaptivity on two different sides. On the one hand, it implements a coarse-grained adaptivity system, so that the ST application implementation is 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 system (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

Experimental Results

- Figure: Per-Transaction Energy Consumption
- Figure: Throughput/Energy Consumption Ratio
- Figure: Application Throughput

References


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