Sapienza Università di Roma
Department of Computer, Control, and Management Engineering

Adaptive Transactional Memories: Performance and Energy Consumption Trade-offs

Diego Rughetti, Pierangelo Di Sanzo, Alessandro Pellegrini

DIAG – Sapienza, University of Rome

Concurrency Control in TMs

Optimistic transaction execution

↓

massive exploitation of available resources (CPU-cores) → generally, better performance than pessimistic (e.g. lock-based) execution

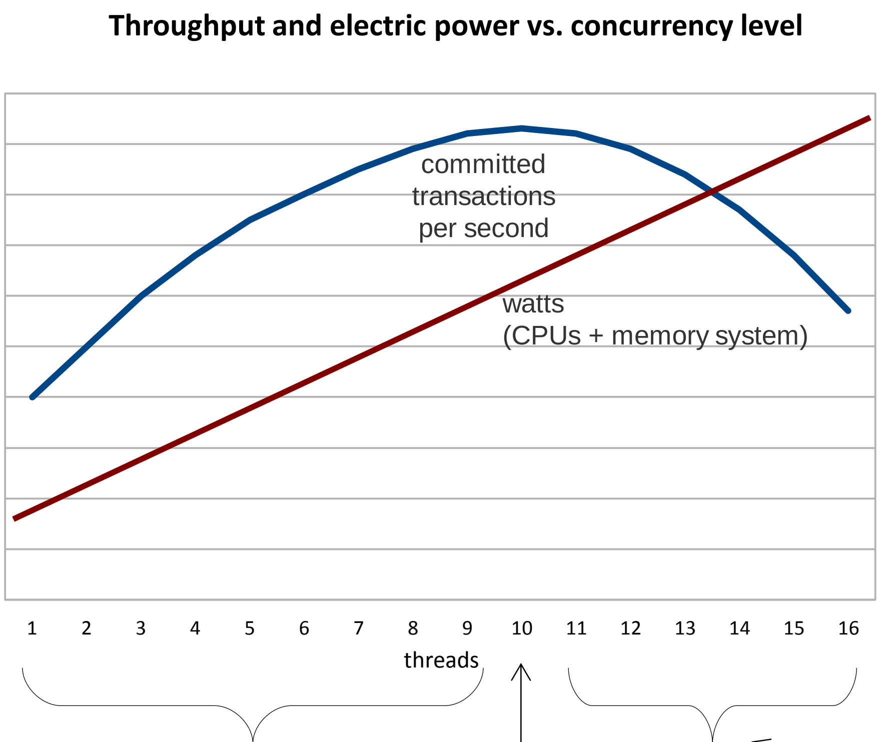
↓

What about energy?

aborted transaction → wasted work → wasted energy
more concurrent threads (more active CPU-cores) → higher transaction abort rate

} more wasted energy

Transactional Memories: How Many Threads?



concurrency level too low: performance is penalized due to limitation of parallelism and underutilization of hardware resources

optimal performance

concurrency level too high: performance loss due to high **data contention** entailing **transaction aborts** and **re-runs**.

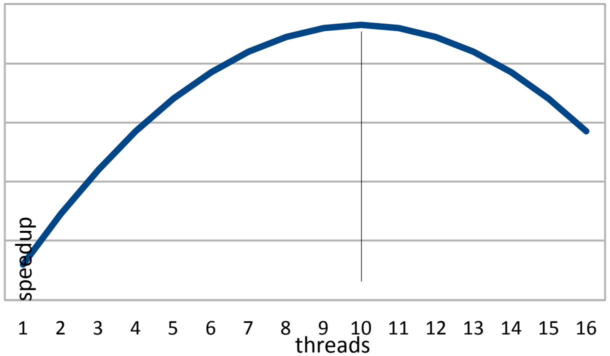
Identifying the optimal concurrency level...

The optimal concurrency depends on:

- application logic
- workload profile
- hardware architecture

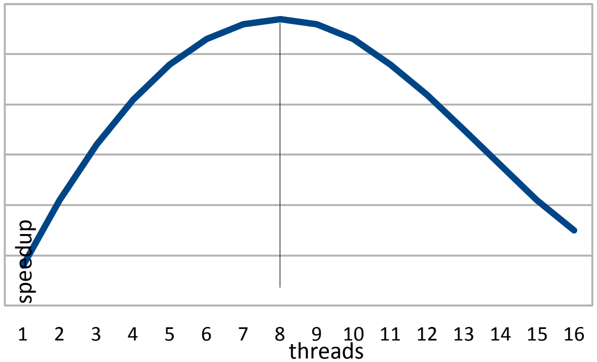
Additionally, the optimal concurrency level may change depending on the **application execution phase**.

phase 1



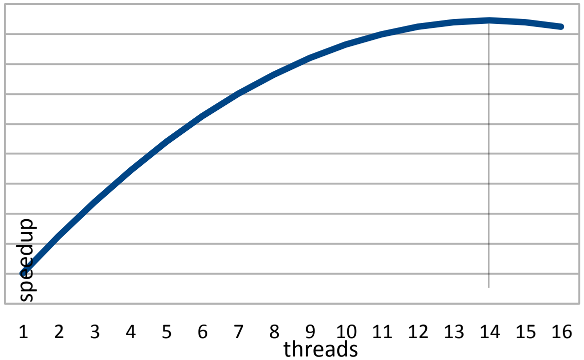
optimal concurrency level: 10

phase 2



optimal concurrency level: 8

phase 3



optimal concurrency level: 14

The study

- Adaptivity in TM implementations can improve performance
- Adaptivity approaches:
    - transaction scheduling
    - thread scheduling
- Performance / energy consumption evaluation study
    - six software transactional memory implementations
    - both transaction and thread scheduling algorithms
    - different scheduling mechanisms
    - different concurrency control algorithms

- Again: what about energy?

Compared STM Implementations

- TinySTM: STM implementation based on Encounter-Time Locking (ETL) algorithm. Used as baseline.
- SAC-STM: adaptive STM implementation based on TinySTM. Thread scheduling based on neural network performance prediction scheme.
- SCR-STM: adaptive STM implementation based on TinySTM. Thread scheduling based on analytic model performance prediction scheme.

Compared STMs

- ATS-STM: adaptive STM implementation based on transaction-scheduling algorithm relying on run-time measurement of the transaction Contention Intensity (CI).

- Shrink: adaptive STM implementation based on transaction-scheduling algorithm relying on temporal locality (basic idea: consecutive transactions executed by a thread access the same data objects).

- R-STM: adaptive STM implementation based on dynamic selection of the concurrency control algorithm.

Experimental Environment

Hardware:

HP ProLiant Server:

- 2 x 8-cores AMD Opteron Processor :16 cores total
- 32 GB RAM
- OS: Linux Debian 6 – kernel 2.7.32-5-amd64

STAMP Benchmarks:

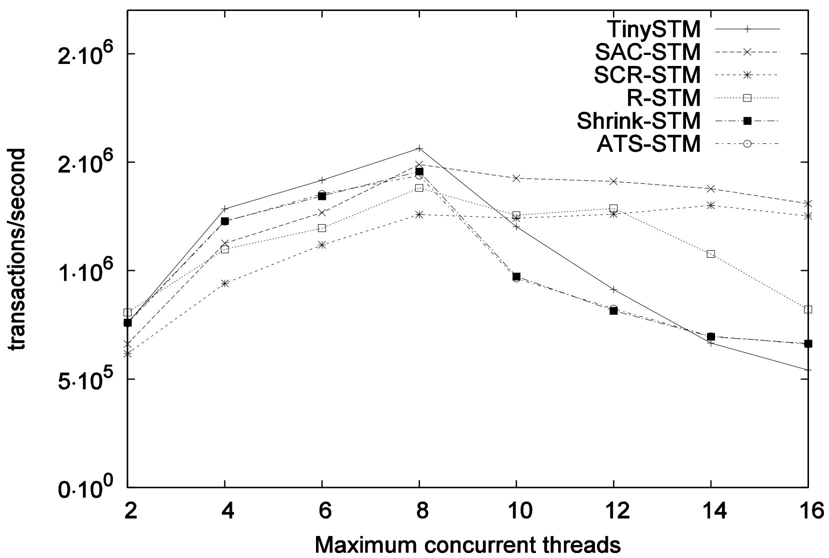
- *intruder* (Network Intrusion Detection System) – Time spent in transactions is relatively moderate
- *yada* (Delaunay Mesh Refinement) – The overall execution time is relatively long, with a high duration of transaction operations and a significantly higher number of memory operations.

Energy consumption measurement:

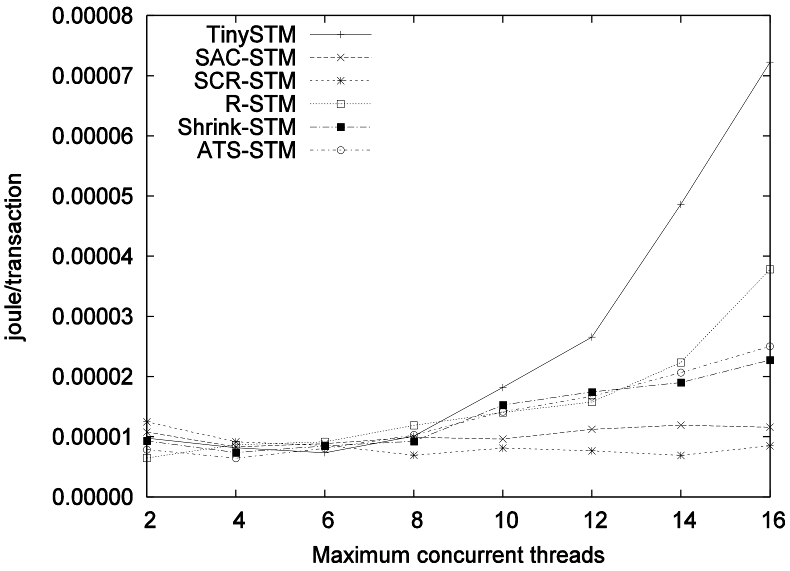
- **pTop** monitoring tool (per-process measurements, exploits Linux kernel Performance Counters management architecture).

Results: Intruder Benchmark

Application throughput

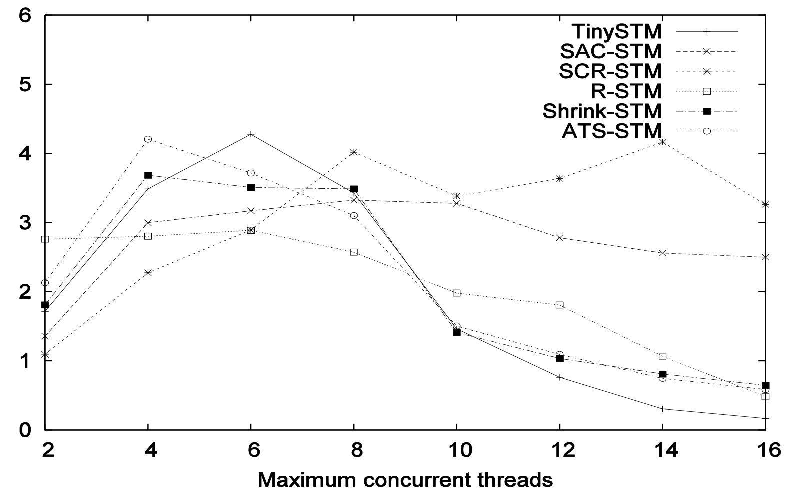


Energy consumption per transaction

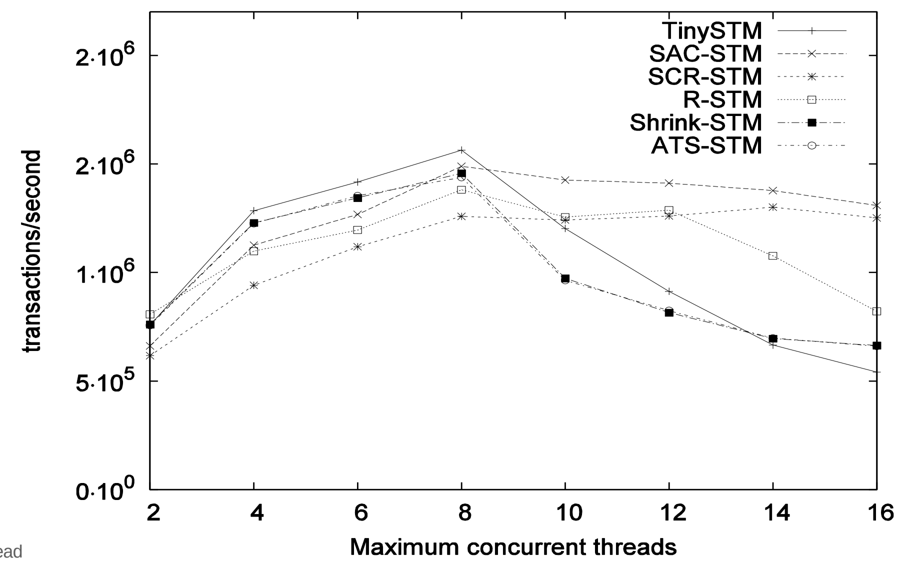


Results: Intruder benchmark

Performance speed up / Energy scaling



Application throughput

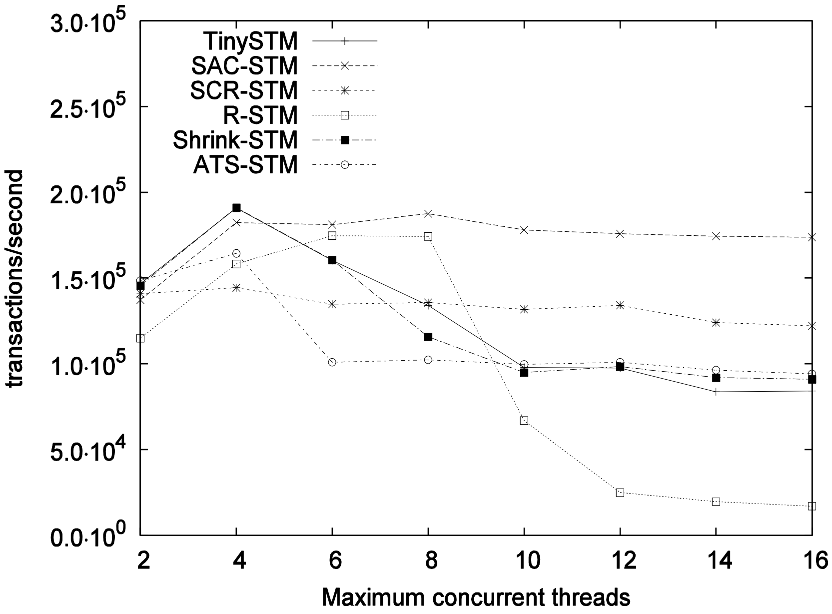


Performance speed up = $\text{throughput}_{k\text{-thread}}/\text{throughput}_{1\text{-thread}}$

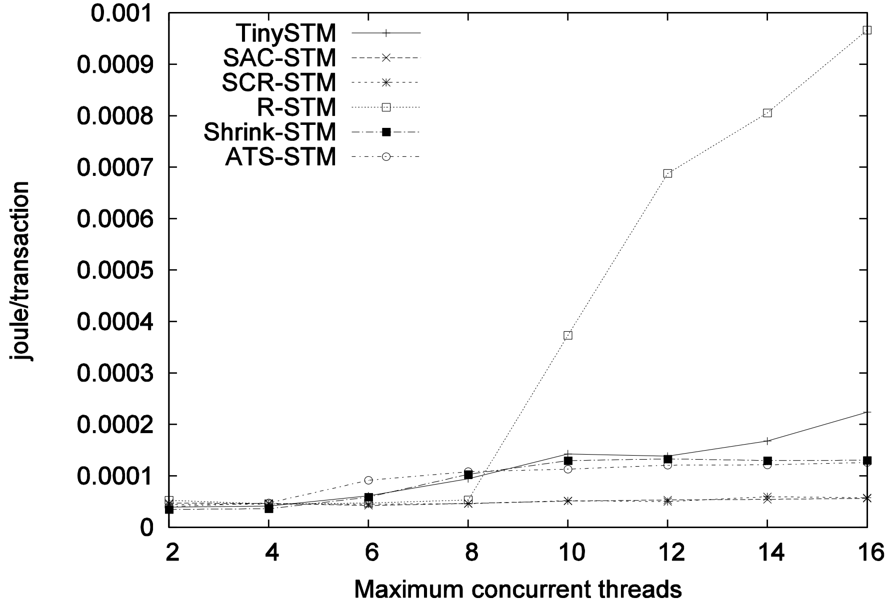
Energy scaling =Joule per transaction $_{k\text{-thread}}$ / Joule per transaction $_{1\text{-thread}}$

Results: Yada benchmark

Application throughput

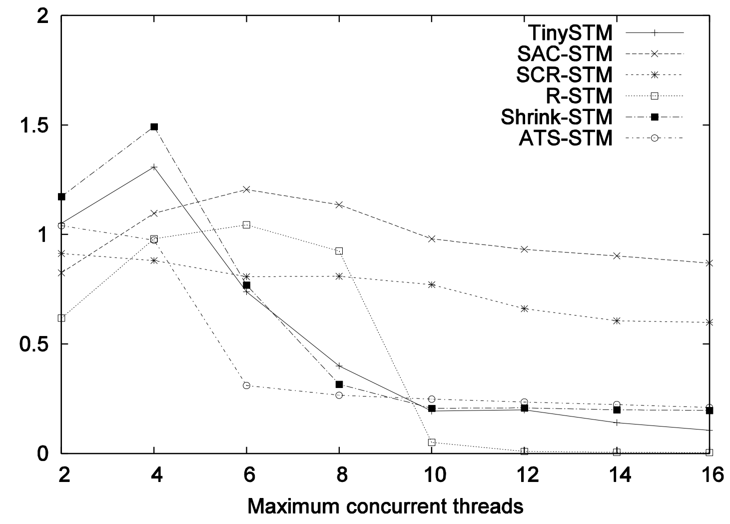


Energy consumption per transaction

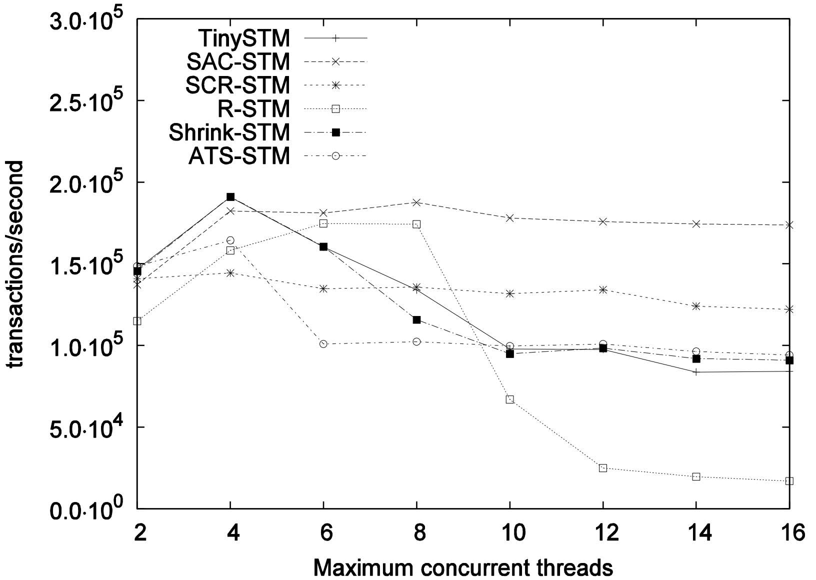


Results: Yada benchmark

Performance speed up / Energy scaling



Application throughput



Summary of Findings

Energy consumption

Less cores than the optimal value:

- Overhead associated to adaptivity mechanisms little affects energy consumption

More cores than the optimal value:

- adaptive transaction/thread scheduling schemes effectively reduce energy consumption
- adaptive concurrency control algorithm selection (R-STM) is not adequate to avoid/reducing energy consumption
- best results are achieved by using application-specific performance schemes

Summary of Findings

Performance vs. Energy consumption

- Extra energy consumption may be required for achieving maximum performance
- Anyway, if we do not really want maximum performance (e.g. SLAs are satisfied with lower performance) a performance/energy trade-off exists:
    - There is a concurrent threads range in which application speed-up increases faster than the energy cost per transaction

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

Thanks for your attention!

Questions?