# Deterministic execution of TM applications

Vesna Smiljković, Christof Fetzer*

Osman Unsal, Adrián Cristal and Mateo Valero

*Barcelona Supercomputing Center, Spain*

**Technische Universität Dresden, Germany*

# Introduction

- Multicore systems and multithreaded applications
- Threads interleave in arbitrary order (nondeterminism)
- Difficult to develop, test, debug
- Difficult to provide fault-tolerance and attack-tolerance
- Determinism:
  - Repeatability
  - Easier finding and solving bugs
  - Easier providing tolerance to faults/attacks in replica-based systems with no communication among replicas

# Deterministic execution

- 1 running thread at a time
- Threads execute in a previously defined order (round-robin)
- Synchronization operations - points where a thread changes its state
- Thread's states: running, ready, blocked
- Providing strong determinism – deterministic execution of code within and outside critical sections/atomic blocks (important for code with data races)
- Implementations:
  - Det-Serial
  - Det-Parallel

# Implementation – Det-Serial

- A runtime C library
- Synchronization operations - points where a thread changes its state:
  - pthread_create, pthread_yield, pthread_join, pthread_barrier_init, pthread_barrier_destroy, pthread_barrier_wait
  - sleep
- Running all code in serial

# Implementation – Det-Parallel

- Based on Det-Serial
- Running non-transactional code in serial
- Transactions:
  - start in serial
  - run in parallel
  - commit in serial
- STM with 2 conflict detection policies (Lazy, Eager)
- Det-Parallel:
  - Lazy Det-Parallel
  - Eager Det-Parallel

# Implementation – Det-Parallel

### Lazy Det-Parallel

- Transactions:
  - start in serial
  - run in parallel
  - commit in serial (conflict detection and aborts)

Additional barriers to synchronize threads

### Eager Det-Parallel

- Transactions:
  - start in serial
  - run in parallel (conflict detection and aborts)
  - commit in serial

Additional barriers to synchronize threads

# Implementation – Det-Parallel(2)

**Eager Det-Parallel**

- Transactions:
  - start in serial
  - run in parallel (conflict detection and aborts)
  - commit in serial

**Step 1.**
- the first-to-commit transaction never aborts
- other conflicting transactions abort

**Step 2.**
- provide fairness
- the first-to-commit thread commits consecutive txns

Fewer barriers to synchronize threads ✓

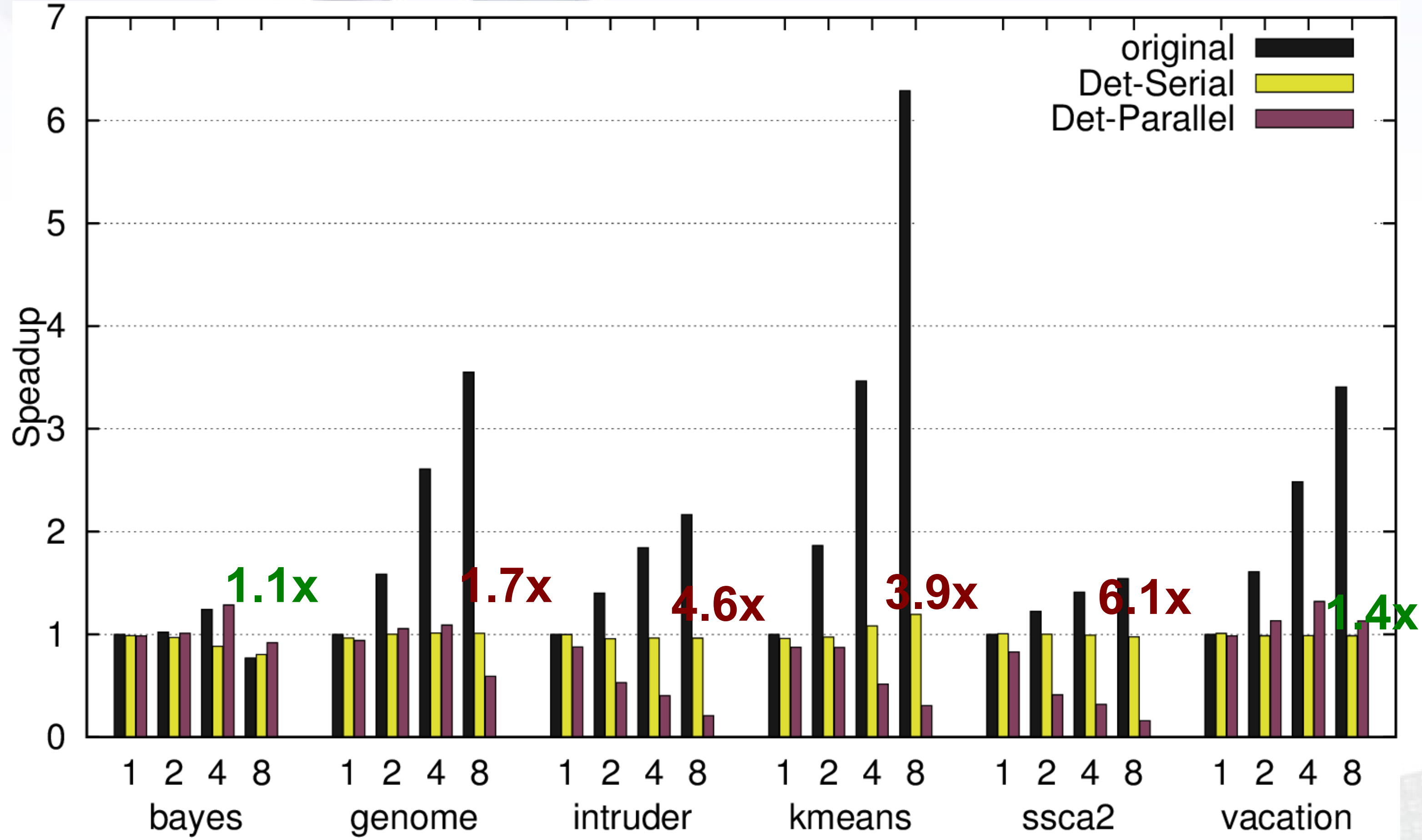
# Evaluation

- Environment:
  - 2 Intel Xeon E5405 quad-core processors (8 cores in total),
  - 4GiB RAM
- TM support:
  - TinySTM 1.0.3.
- Benchmarks
  - STAMP [1]

[1] Minh, C., Chung, J., Kozyrakis, C., Olukotun, K.: STAMP: Stanford transactional applications for multi-processing. In: Workload Characterization, IISWC 2008.

BSC Microsoft Research Centre

# Evaluation - STAMP




Det-Parallel vs. Det-Serial
(8 threads)

Speedup: Bayes, Vacation

Slowdown: Genome, Intruder, Kmeans, SSCA2

BSC Microsoft Research Centre

# Conclusions

- Determinism:
  - to find and solve bugs easier
  - to provide fault- and attack- tolerance easier
- Different implementations of a deterministic system
- Provide deterministic and parallel execution of transactions
- Additional synchronization
  - provides strong determinism
  - lowers the benefit of parallel execution of transactions

# Future work

- Eager Det-Parallel
  - to ensure progress of the first-to-commit transaction
  - to reduce the number of synchronization barriers
- Comparison with DTHREADS [2]
- Various schedule algorithms
- Determinism and diversity

[2] T. Liu, C. Curtsinger, and E. D. Berger. DTHREADS: efficient deterministic Multithreading. In Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP '11).

# Thank you for your attention!

vesna.smiljkovic@bsc.es