

EazyHTM: Eager-Lazy Hardware Transactional Memory

Saša Tomić, Cristian Perfumo, Chinmay Kulkarni,
Adrià Armejach, Adrián Cristal, Osman Unsal,
Tim Harris, Mateo Valero

*Barcelona Supercomputing Center, UPC
BITS Pileri
Microsoft Research Cambridge*

Why Transactional Memory?

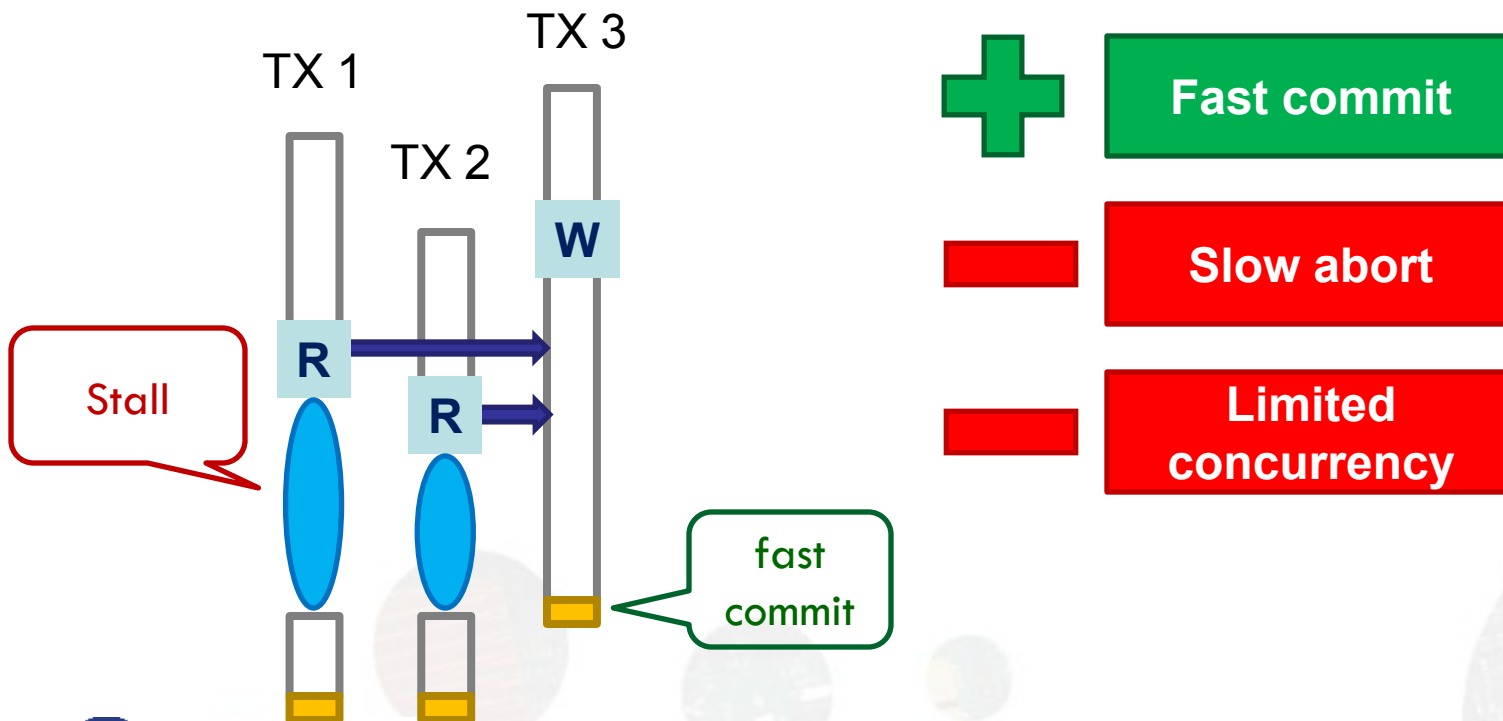
- Lock-based parallel programming has problems
 - Deadlocks, races, complexity, performance, ...
- Transactional Memory (TM) to the rescue
 - Optimistic concurrency control mechanism
 - Easy to use
 - Deadlock free
 - Supports composability
 - Protects data in critical sections
- **Hardware-TM (HTM)**, Software-TM (STM) and hybrid

HTM terminology

- Atomic section/**transaction**: group of instructions that appear to take effect instantaneously
- Where are speculative values stored (version management):
 - in-place, and log the original value, or
 - buffered in private storage, publish on **commit**
- **Conflict**: TX writes where others TX reads
 - **Detection**: an action in which we check for conflicts
 - **Resolution**: an action performed to resolve the conflict
 - Can be abort, stalling the execution, ...

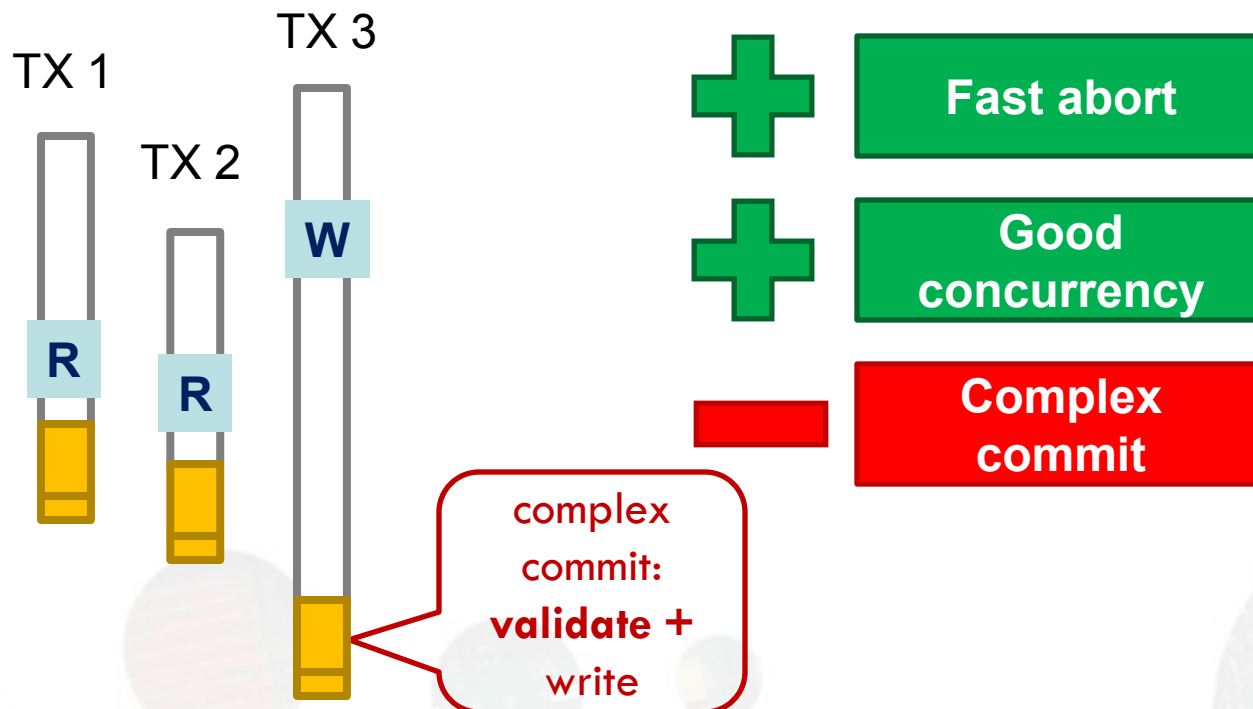
Eager HTM

- A.k.a. pessimistic
- Writes in-place, detects&resolves conflicts on every access
- LogTM [Moore, HPCA06], LogTM-SE [Yen, HPCA07]



Lazy HTM

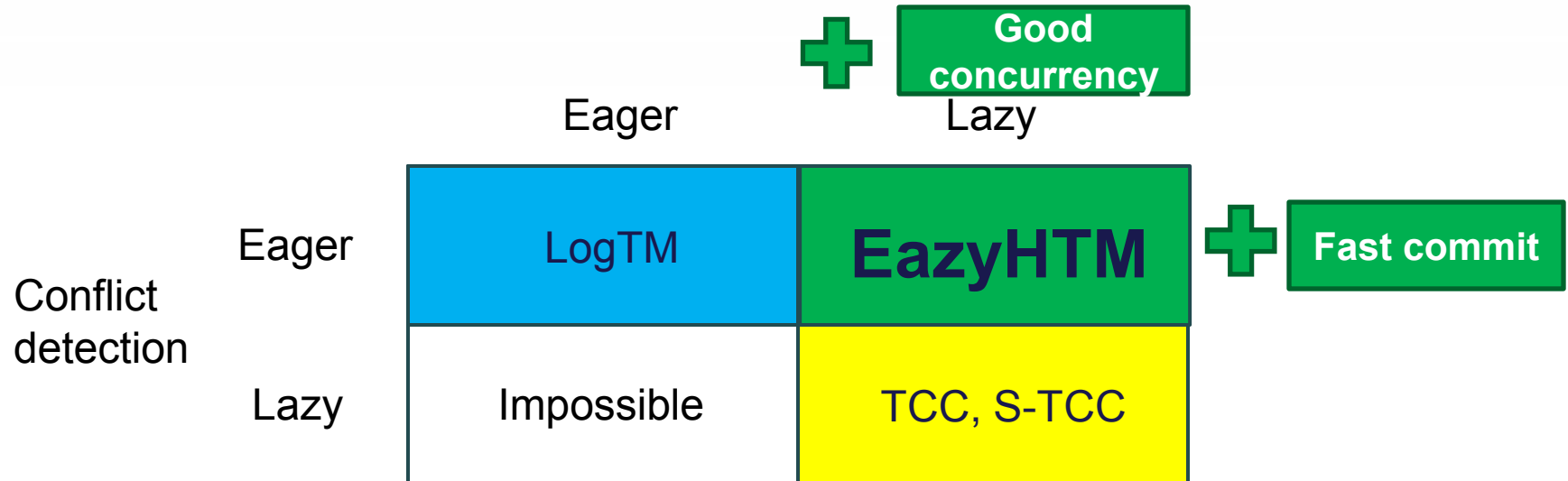
- A.k.a. optimistic
- Writes buffered, detect&resolve conflicts on commit
- TCC [Hammond, ISCA04], Scalable-TCC [Chafi, HPCA07]



The Motivation

Splitting conflict management

Conflict resolution



- Eager-Lazy hardware-software TM exists (FlexTM [Shriraman, ISCA08]):
 - Software begin, commit and abort
 - Probabilistic (signature based) conflict detection
- EazyHTM is the first pure-hardware TM

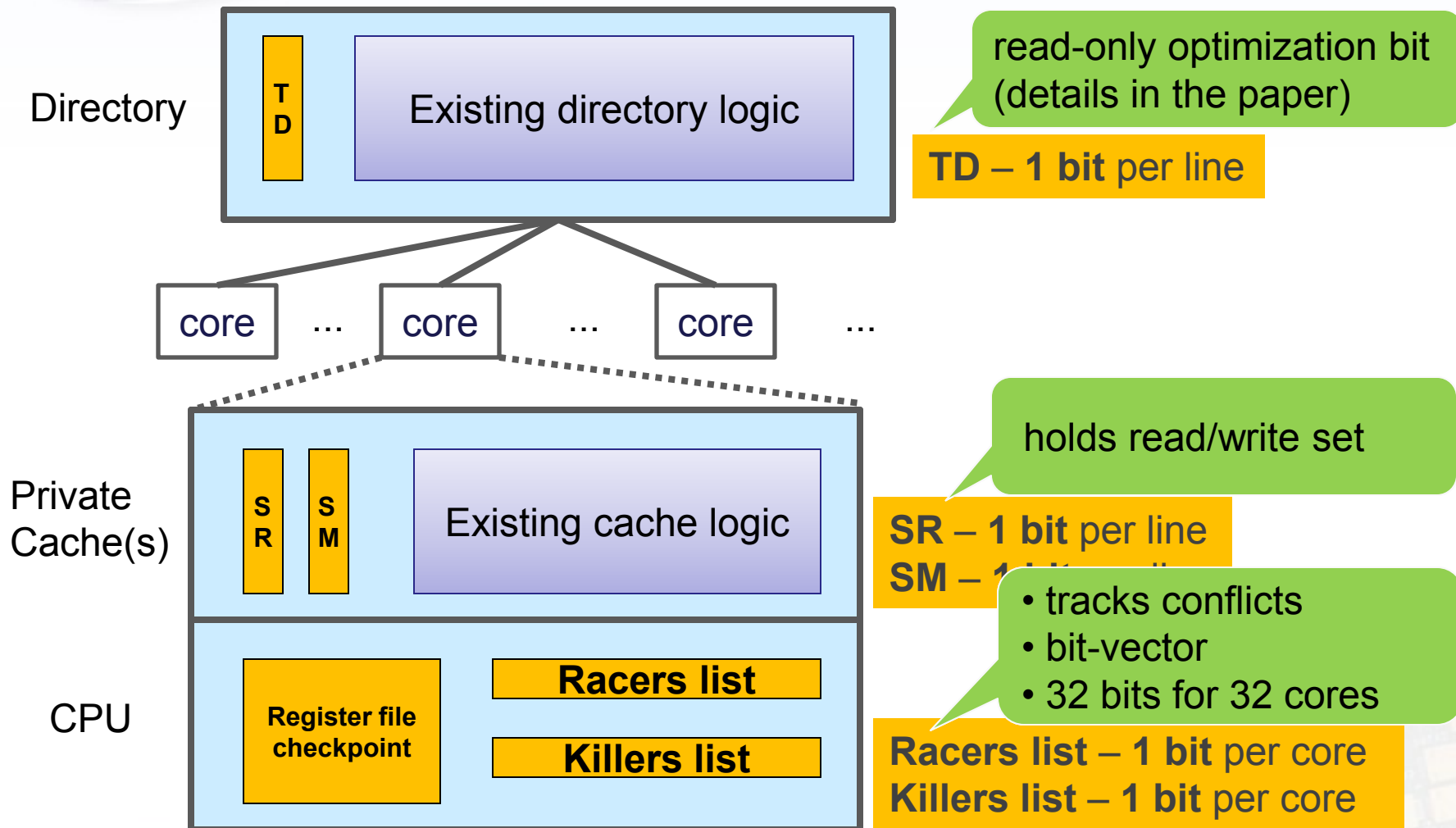
Outline

- Motivation
- Contributions
- Hardware changes
- The Protocol
- Evaluation
- Conclusions

EazyHTM Contributions

- The best of two worlds
 - Eager conflict detection: **simple commit/exact list of conflicts in advance**
 - Lazy conflict resolution: **good concurrency**
- Parallel commits of non-conflicting TXs
- Designed for CMPs (Chip-Multiprocessors)
 - Use cores **proximity**
 - MESI/MOESI protocol upgrade (easier verification)

Hardware changes

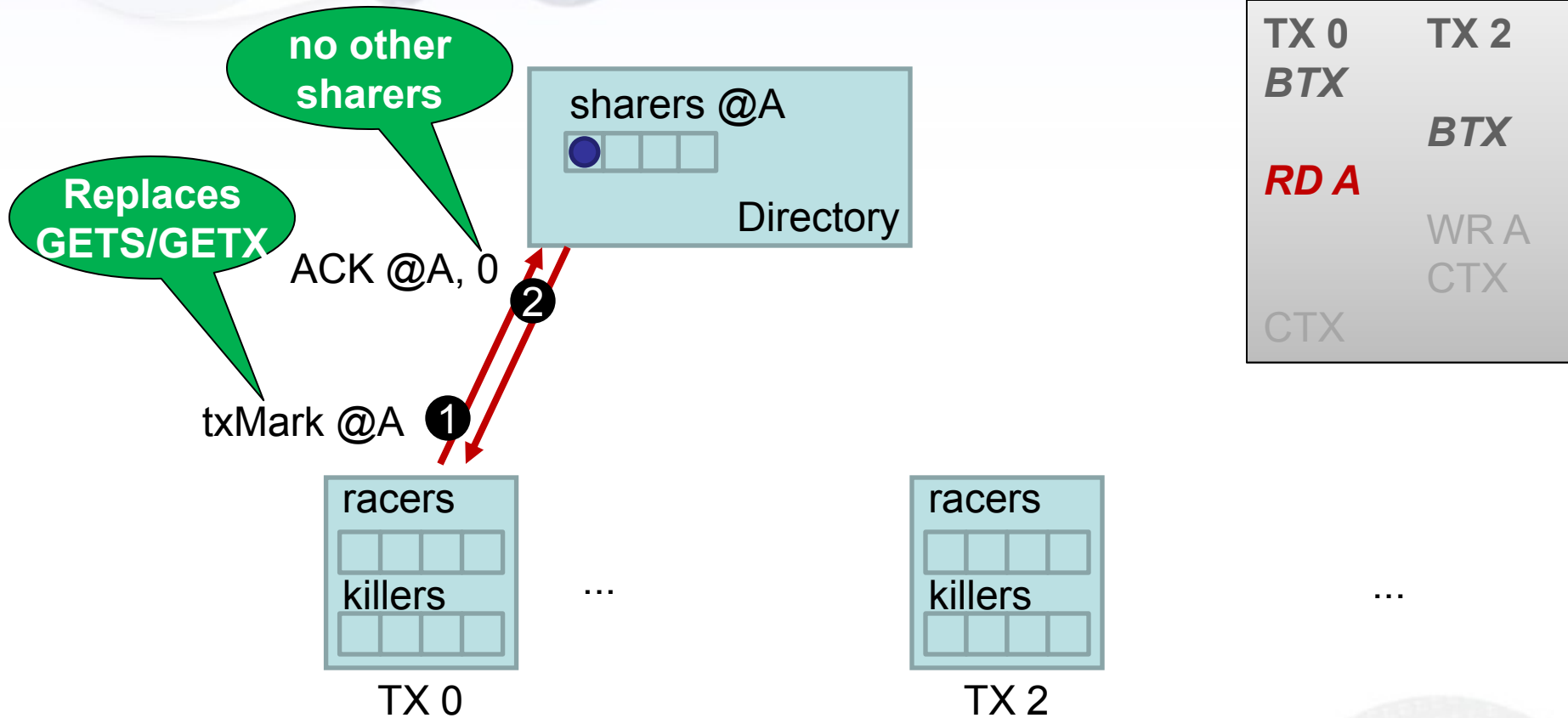


Racers and killers list

- If line is shared between two TXs:
 - Read-Read
 - No conflict
 - Write-Read, Read-Write, Write-Write
 - Writer adds reader TX into **“racers” list**
 - **“TXs that I have to abort” list, if I commit first**
 - Reader adds writer TX into **“killers” list**
 - **“TXs that can abort me” list, if they commit first**
- We illustrate only the Write-after-Read (WAR) conflict

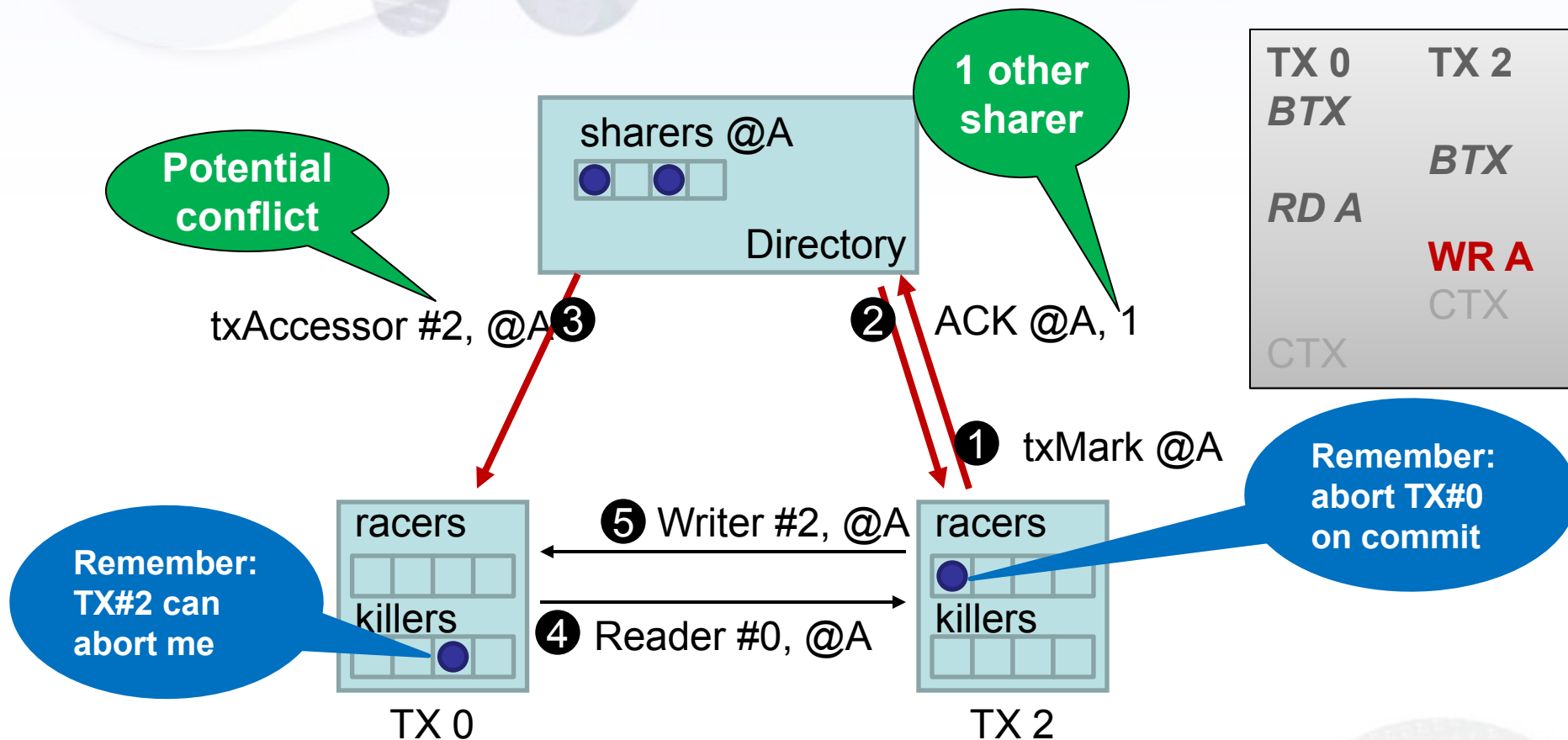
EazyHTM Protocol

Conflict Detection (1/2)



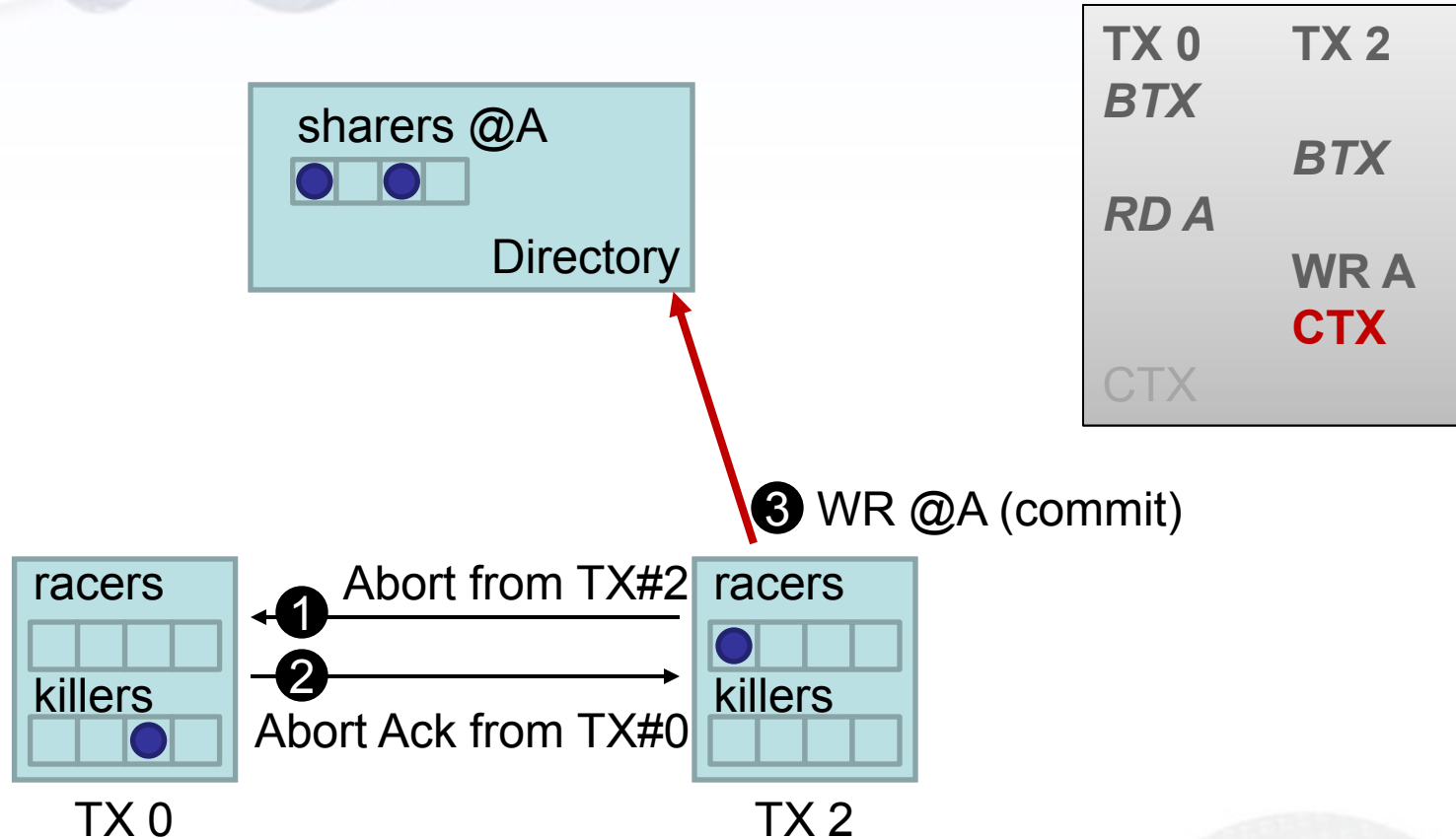
EazyHTM Protocol

Conflict Detection (2/2)



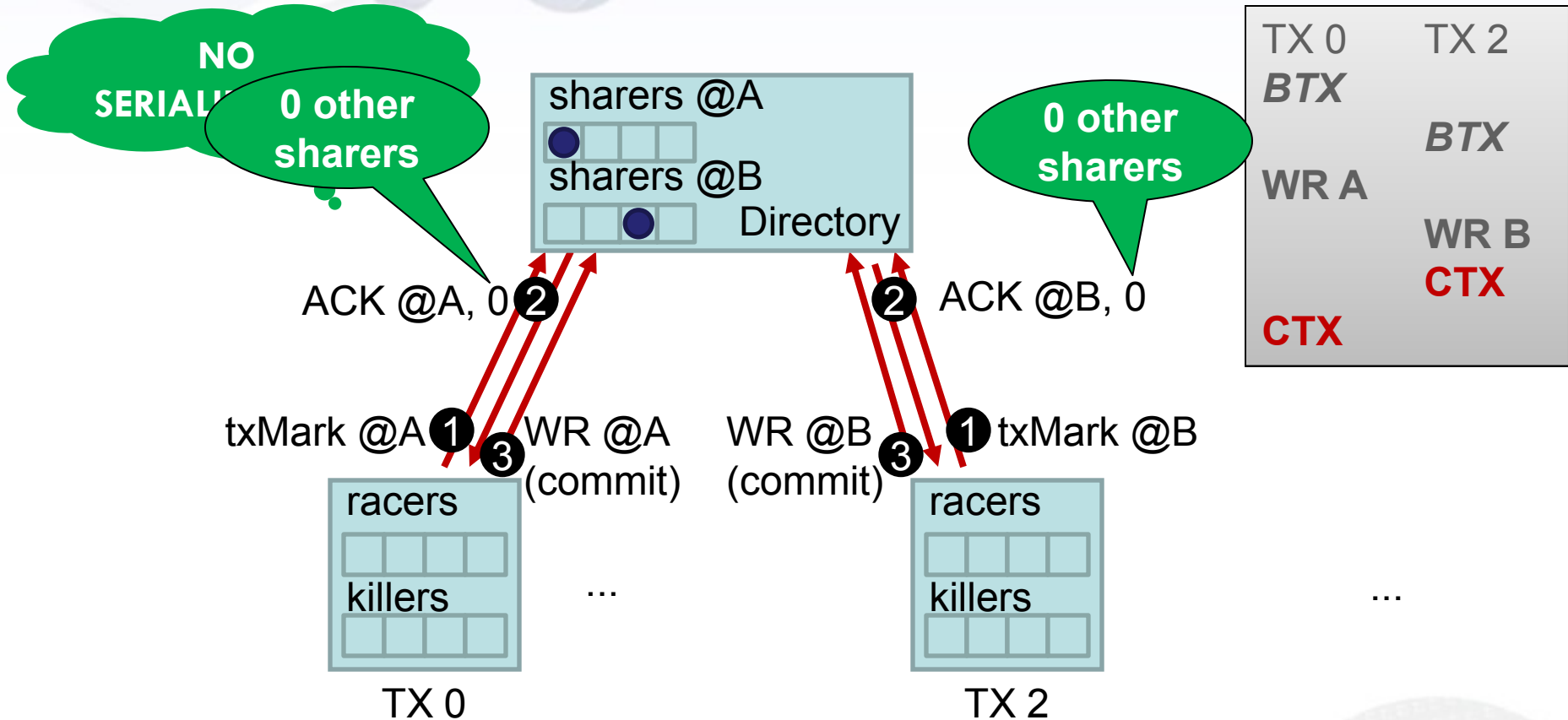
EazyHTM Protocol

Conflict Resolution



- 1 TX#2 first came to the commit point, abort TX#0!

Disjoint data => parallel commit



TX#0 works with line @A

TX#2 works with line @B

Implementation

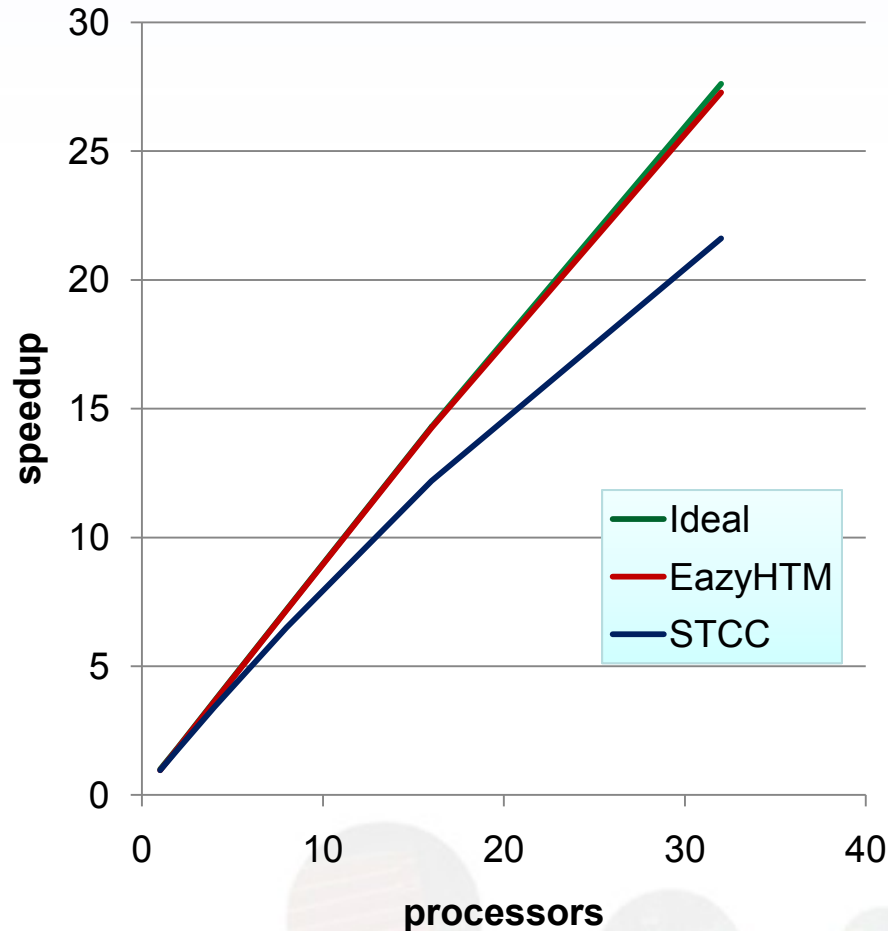
- Implemented in M5, full-system simulator (Alpha)
- Private L1 (32KB, 4-way, 64B CL, 2 cycles)
- Private L2 (512KB, 8-way, 64B CL, 10 cycles)
- Memory (with directory, 100 cycles)
- ICN (2D Mesh, 10 cycles per hop)

Evaluation

- Evaluated STAMP benchmarks
- Compared **with Scalable-TCC-like HTM**
 - Same base simulator
 - Implemented specialized directory protocol
- Compared **with ideal lazy HTM** (MESI based)
 - magical conflict detection
 - instant conflict resolution
 - parallel write-back commit

Kmeans Low

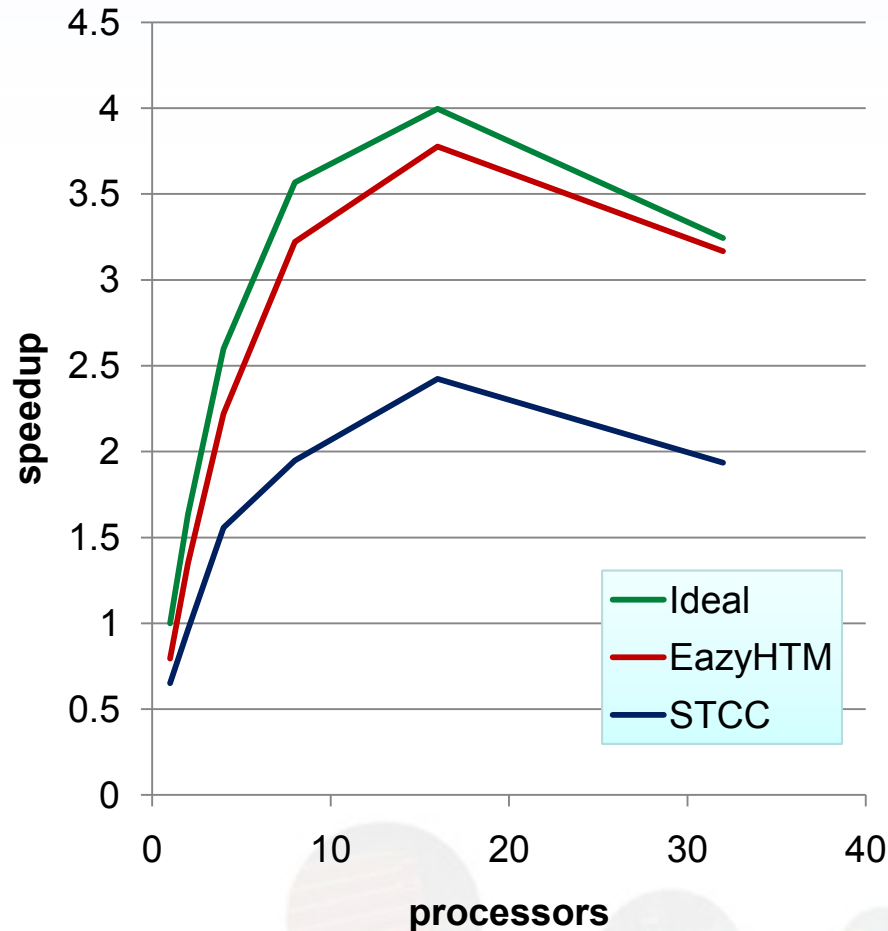
Kmeans-Low



- **Small TXs** (RS 15 CL; WS 5 CL)
- **Low contention (10% aborts)**
- Similar profile to “replacing locks with atomic”
- Near ideal performance
- K-means: groups N-dimensional space into K clusters
- Most of the **SPLASH-2 suite** has similar profile

SSCA2

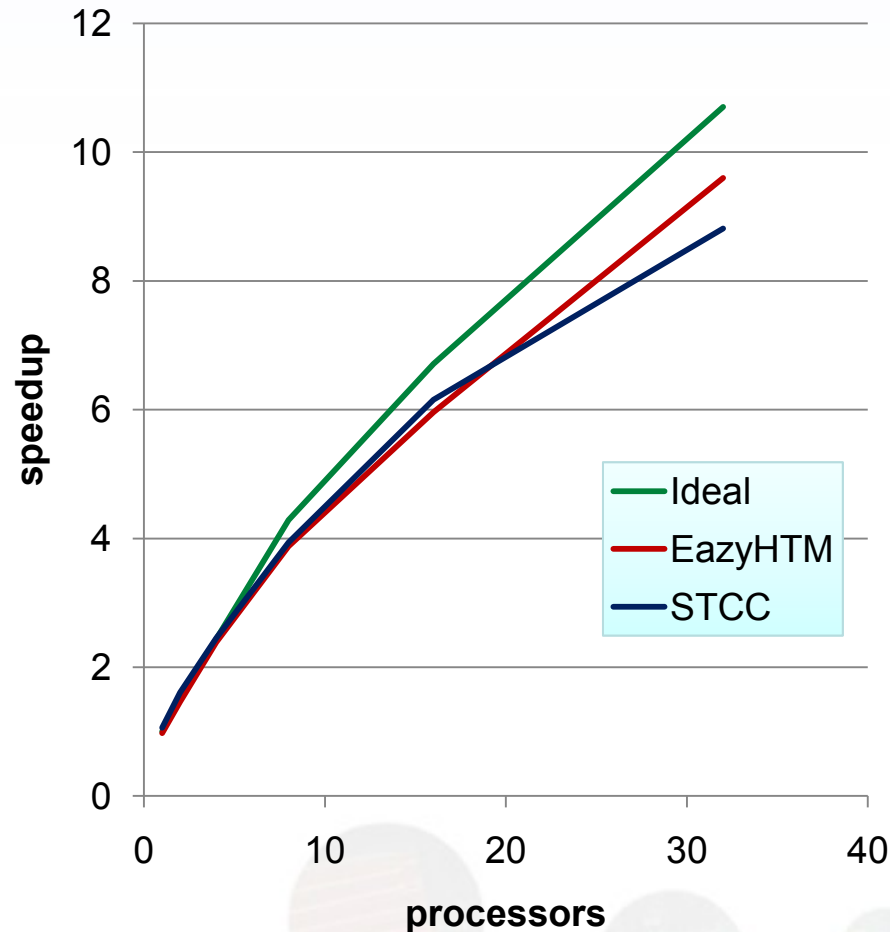
SSCA2



- **Small TXs** (RS 50 CL, WS 10 CL)
- **Low contention** (**1.2% aborts**)
- Near ideal performance
- Scalability affected by barriers, not by contention
- SSCA2: large directed graph operations

Yada

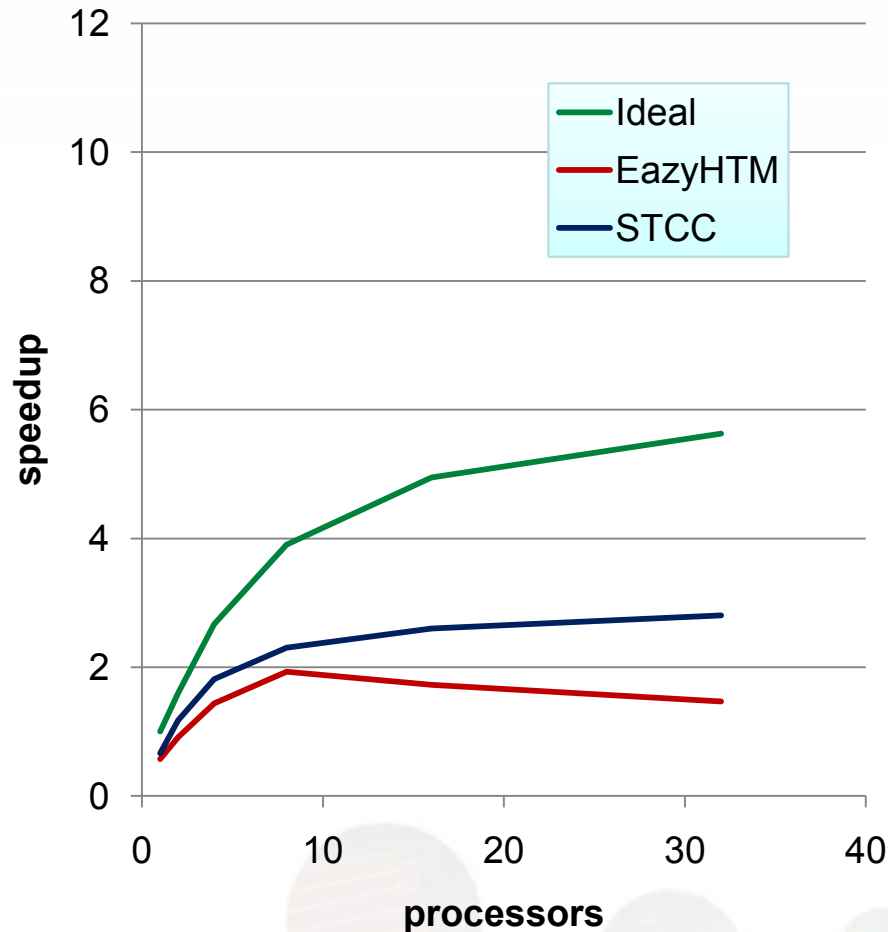
Yada



- **Large TXs** (260 CL RS, 140 CL WS)
- **Moderate contention** (**35% aborts**)
- We can see good performance also for large TXs!
- Yada: delaunay mesh refinement

Intruder

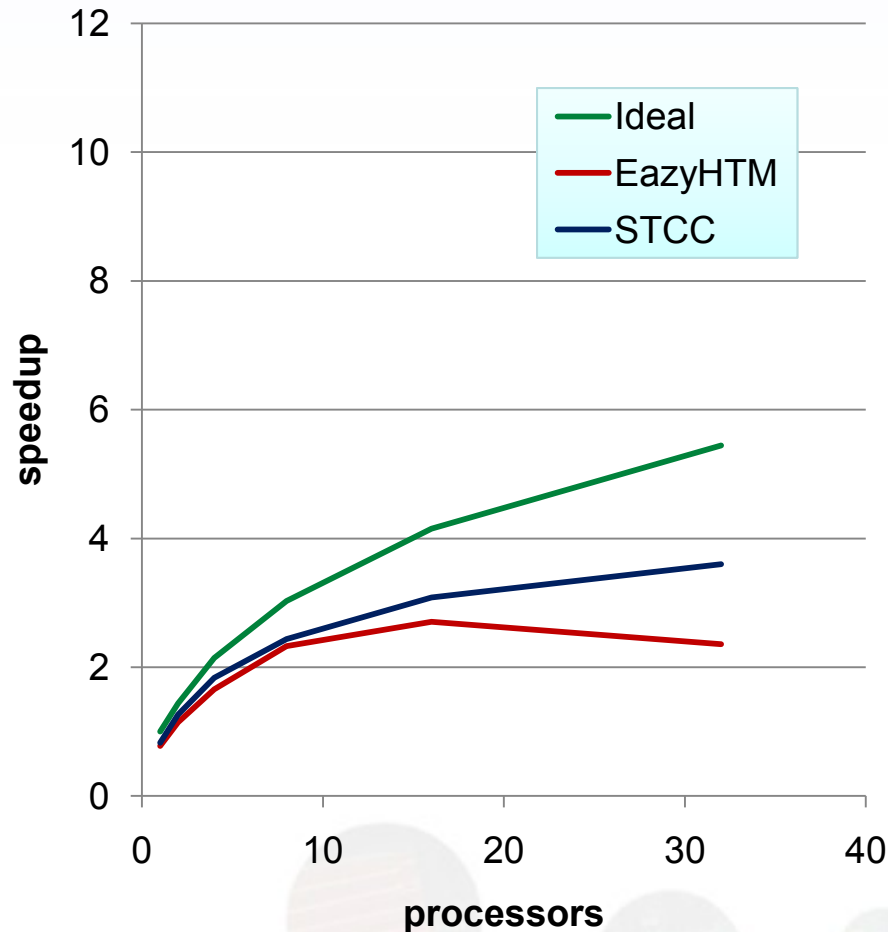
Intruder



- **Medium TXs** (53 CL RS, 20 CL WS)
- **High contention (85% aborts)**
- Very bad scalability for all HTMs
- Every transaction detects conflicts over and over again – lot of conflict detection messages slow down the execution
- Intruder: signature based network intrusion detection system

Only high-conflict STAMP

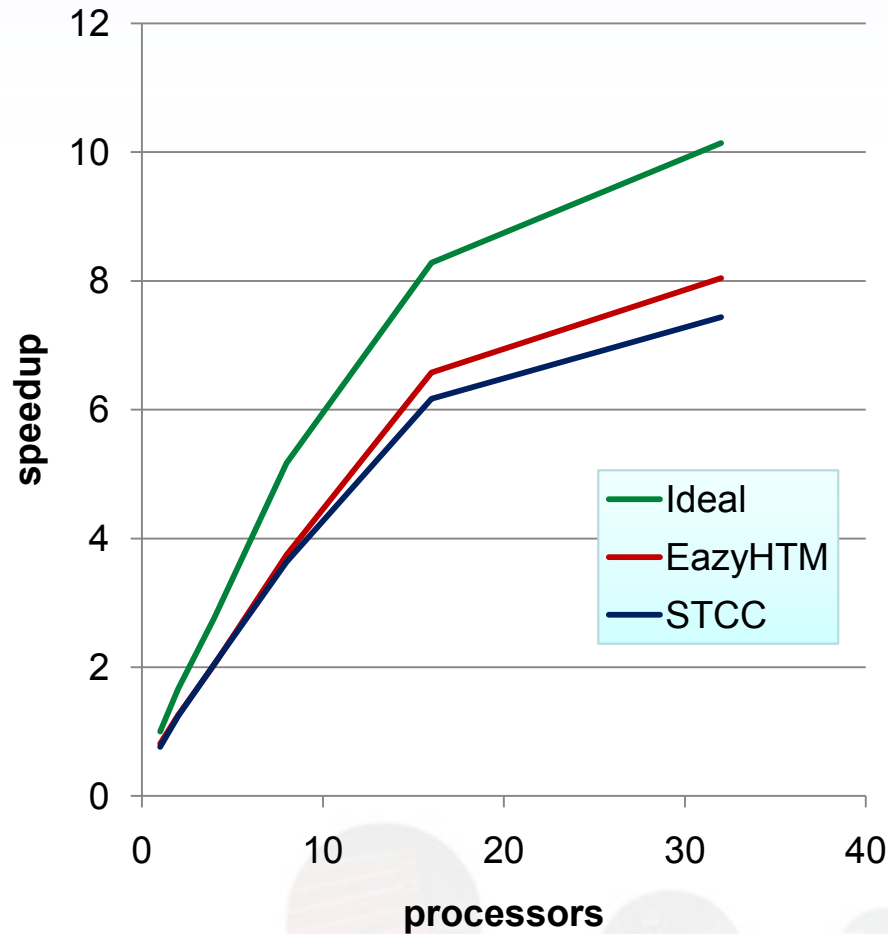
High-conflict STAMP



- **>50% abort rate only**
- High contention high-core-count should be optimized
- Averages:
 - Labyrinth
 - Intruder
 - Kmeans-Hi
- Results highly affected by Intruder

Only low-conflict STAMP

Scaling STAMP



- **<50% abort rate only**
- Low abort rate **necessary** for scaling
- Excludes:
 - Labyrinth 8-32
 - Intruder 16-32
 - Kmeans-Hi 32

Conclusions

- Introduced EazyHTM, a new HTM implementation
 - **E**ager conflict detection, **l**azy conflict resolution
 - Fast: performs well for low conflict parallel applications
 - Minimal changes to directory protocols (easier verification)
 - As scalable as standard directory protocol
- EazyHTM mechanism could allow (future work):
 - Simpler transaction prioritization
 - Less wasted work
 - Better performance optimization
 - Power efficient TM mechanisms

Thank you!

Questions?

sasa.tomic@bsc.es