FASTLANE
Streamlining Transactions for Low Thread Counts

Jons-Tobias Wamhoff
Christof Fetzer
Technische Universität Dresden, Germany

Pascal Felber
Etienne Rivière
Université de Neuchâtel, Switzerland

Gilles Muller
INRIA, France
Motivation

- Number of cores: 1
- Performance: Slower
- Expected gains from FastLane
- Best performance
- STM
- FastLane
- Sequential
- Many

Number of cores

Performance

Slower

Faster
General Idea

• 1 master thread
  • Commits transactions without aborting
  • Minimal instrumentation and bookkeeping
• N helper threads
  • Commit transactions only when not in conflict
  • Contribute progress without impairing on the performance of the master
Code Paths

START

pessimistic code paths

SEQUENTIAL
  uninstrumented

MASTER
  lightweight instrumented writes

HELPER
  instrumented, synchronize with master

STM
  instrumented, extensive bookkeeping

speculative code paths

COMMIT
Code Paths

- Dresden TM Compiler
  - Generates multiple code paths for sequential (uninstrumented), FastLane (master & helper) and STM
  - Generic `START` and `COMMIT` calls with internal branch
  - `READ` and `WRITE` are specific to code path and inlined
    - transaction descriptor only accessed if needed
- TinySTM++ TM runtime
  - Dynamically select code path based on core or thread count at `BEGIN`
Data Structures

Counter
odd: owned
even: otherwise

Memory

Dirty array

Address written

Timestamp

Address read

Start timestamp

Address read

Timestamp

... Write-set

Address read

... Read-set

Count

odd: owned
even: otherwise

Master thread

isMaster

Helper thread

...
Master vs. Helper

**MASTER**

```
BEGIN
acquire(cntr)

READ (addr)
return *addr

WRITE (addr, val)
dirty[hash(addr)] = cntr
addr = val

COMMIT
release(cntr)
```

**HELPER**

```
BEGIN
start = cntr

READ (addr)
dirty[hash(addr)] ≤ start
add(read-set, addr)
return *addr

WRITE (addr, val)
dirty[hash(addr)] ≤ start
put(write-set, addr, val)

COMMIT
```
3 Commit Variants

COMMIT 1

acquire\((cntr)\)

VALIDATE

abort

proceed
3 Commit Variants

COMMIT 1

acquire(cntr)

VALIDATE

aborted

proceed

COMMIT 2

c = awaitEven(cntr)

VALIDATE

aborted

acquire(cntr)

\[ cntr \leq c+1 \lor \]

VALIDATE

aborted

proceed
3 Commit Variants

**COMMIT 1**
- acquire($cntr$)
- VALIDATE
- abort
- proceed

**COMMIT 2**
- $c = \text{awaitEven}(cntr)$
- VALIDATE
- abort
- proceed
- abort

**COMMIT 3**
- $c = \text{awaitEven}(cntr)$
- VALIDATE
- abort
- failed
- tryAcquire($cntr$, $c$)
- proceed

---
Spear et al.: RingSTM: Scalable Transactions with a Single Atomic Instruction, SPAA '08
Intset Benchmarks
Thank you!