Transactional Tries

Michael Schröder

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Software Transactional Memory

- `atomically { ...` sequential code `... }`
- execute code without taking locks
- record reads & writes in transactional log
- at the end, try to commit effects to memory
- commit may fail → transaction is run again

Software Transactional Memory

in Haskell

```
inc :: TVar Int → STM ()
inc v = do x ← readTVar v
           writeTVar v (x+1)
```

```
atomically $ do inc v1
                inc v2
```

TVar (HashMap k v)

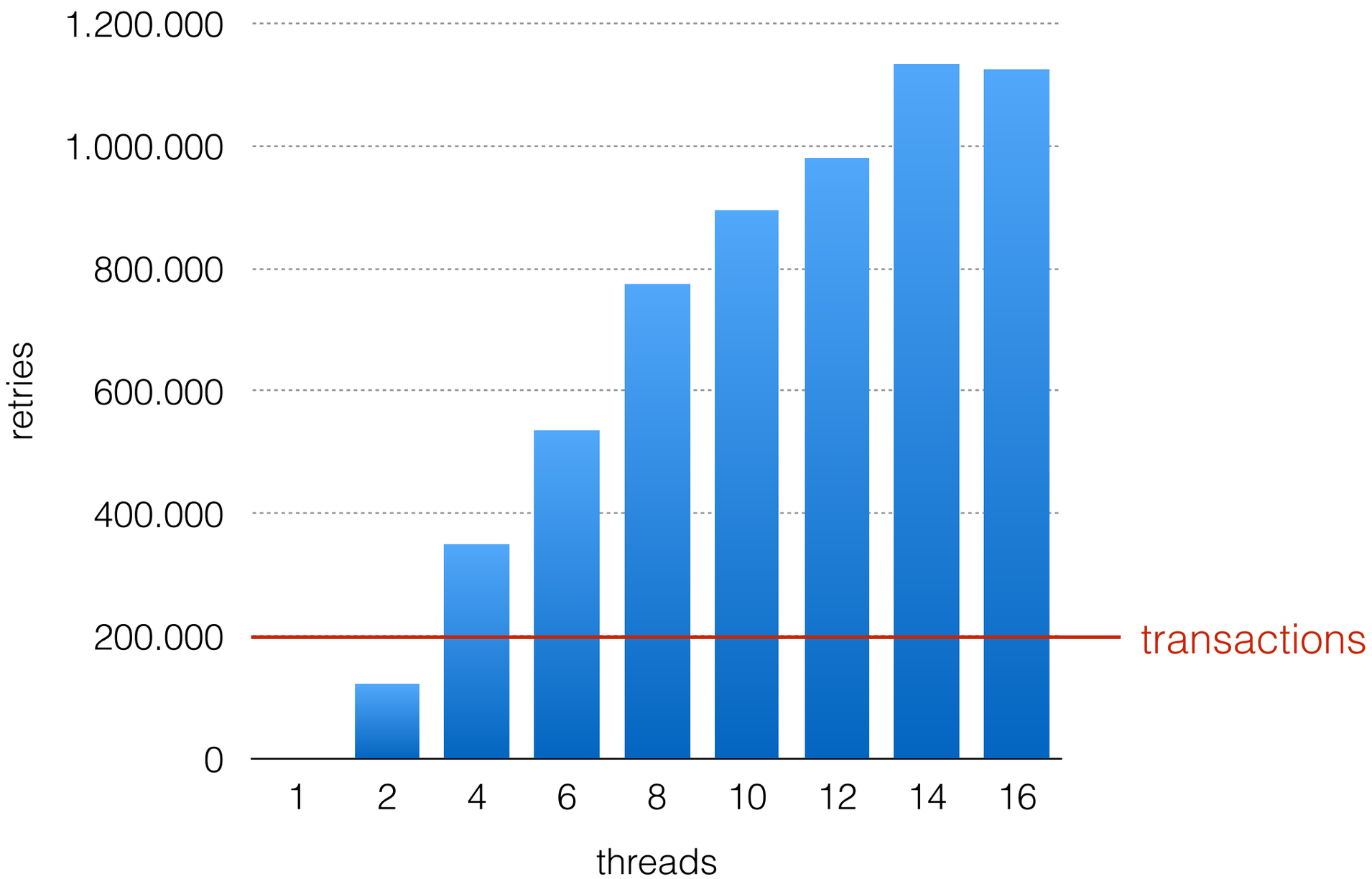
- if transaction *A* updates element k_1
  and transaction *B* deletes element k_2
  and $k_1 \neq k_2$, then there should be no conflict…

- …but there is!

- we have to update the whole container
  and thereby invalidate all other transactions

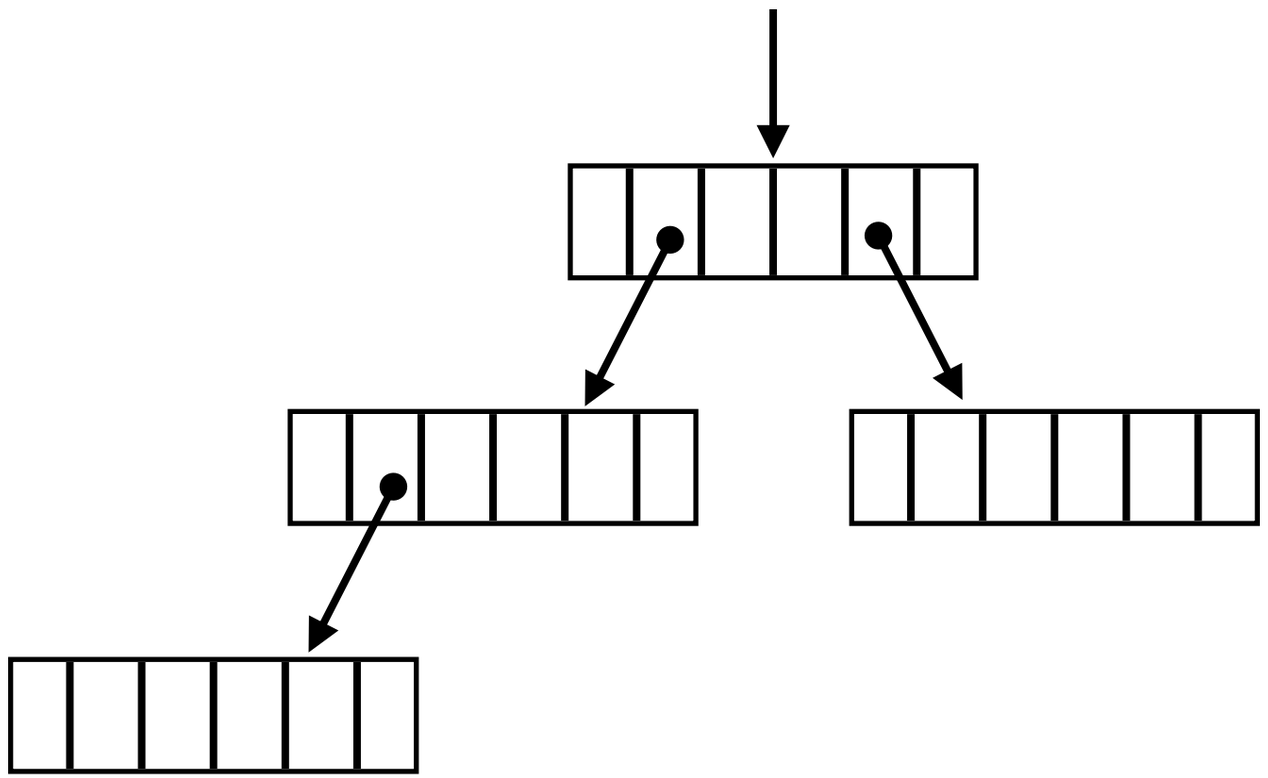
- Problem: **Contention**

TVar (HashMap k v)

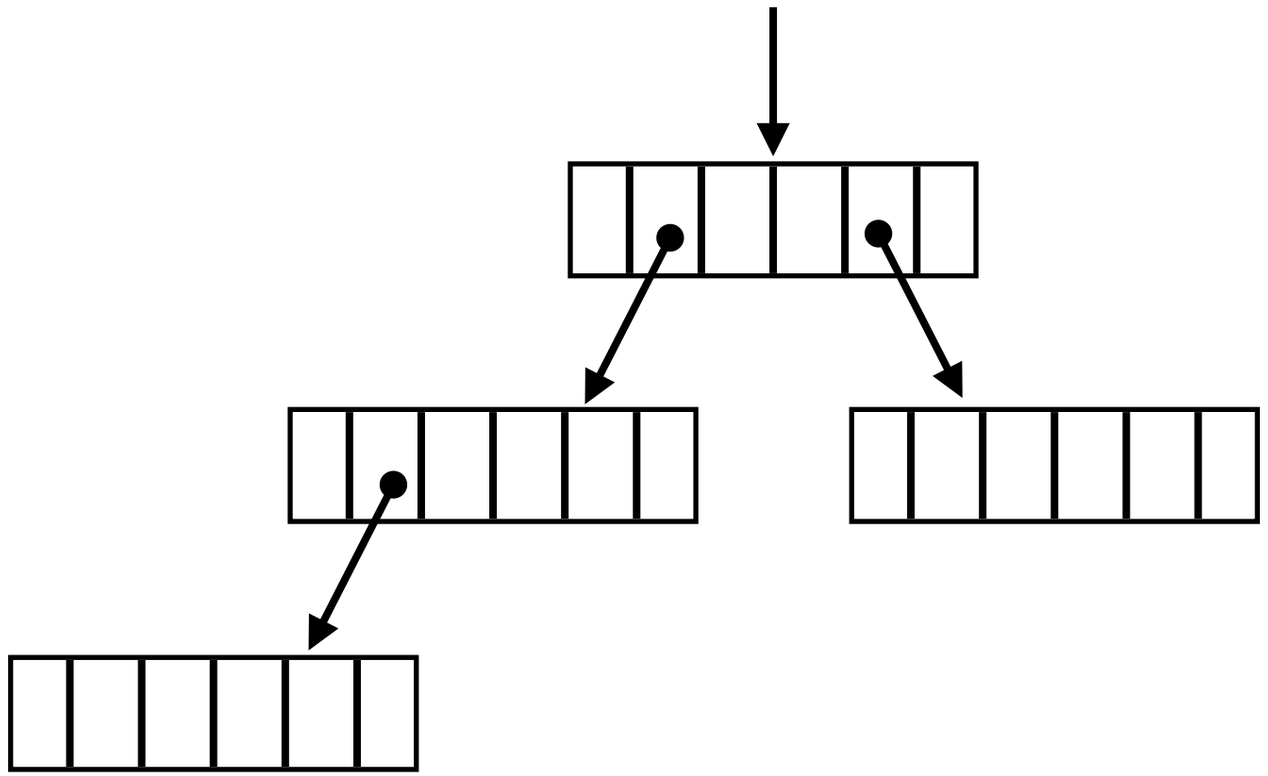


Can we make a hash map for STM that is contention-free?

Hash Trie



Concurrent Trie



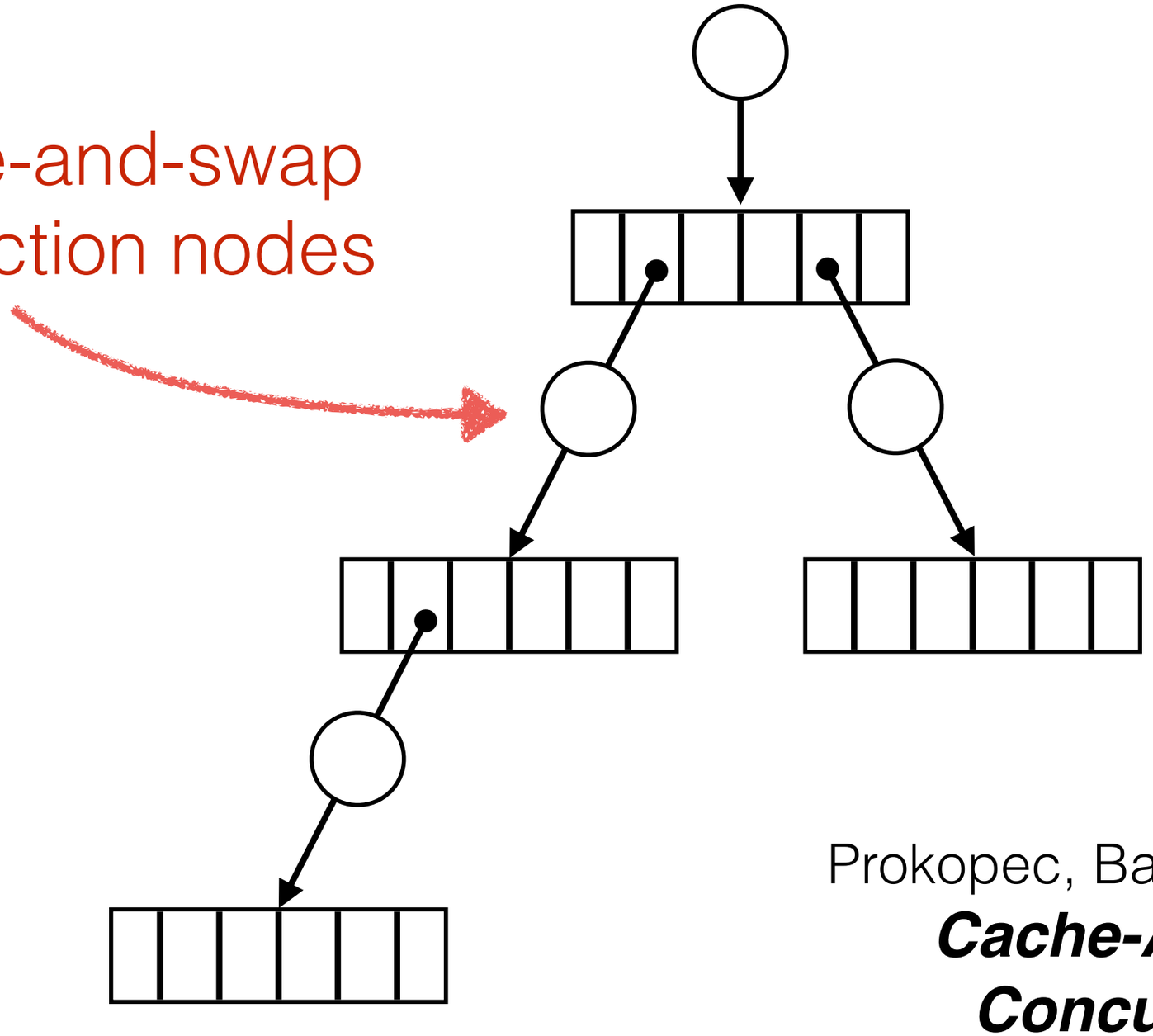
Prokopec, Bagwell, and Odersky

Cache-Aware Lock-Free Concurrent Hash Tries

Tech. Rep. EPFL-REPORT-166908, 2011

Concurrent Trie

compare-and-swap
on indirection nodes



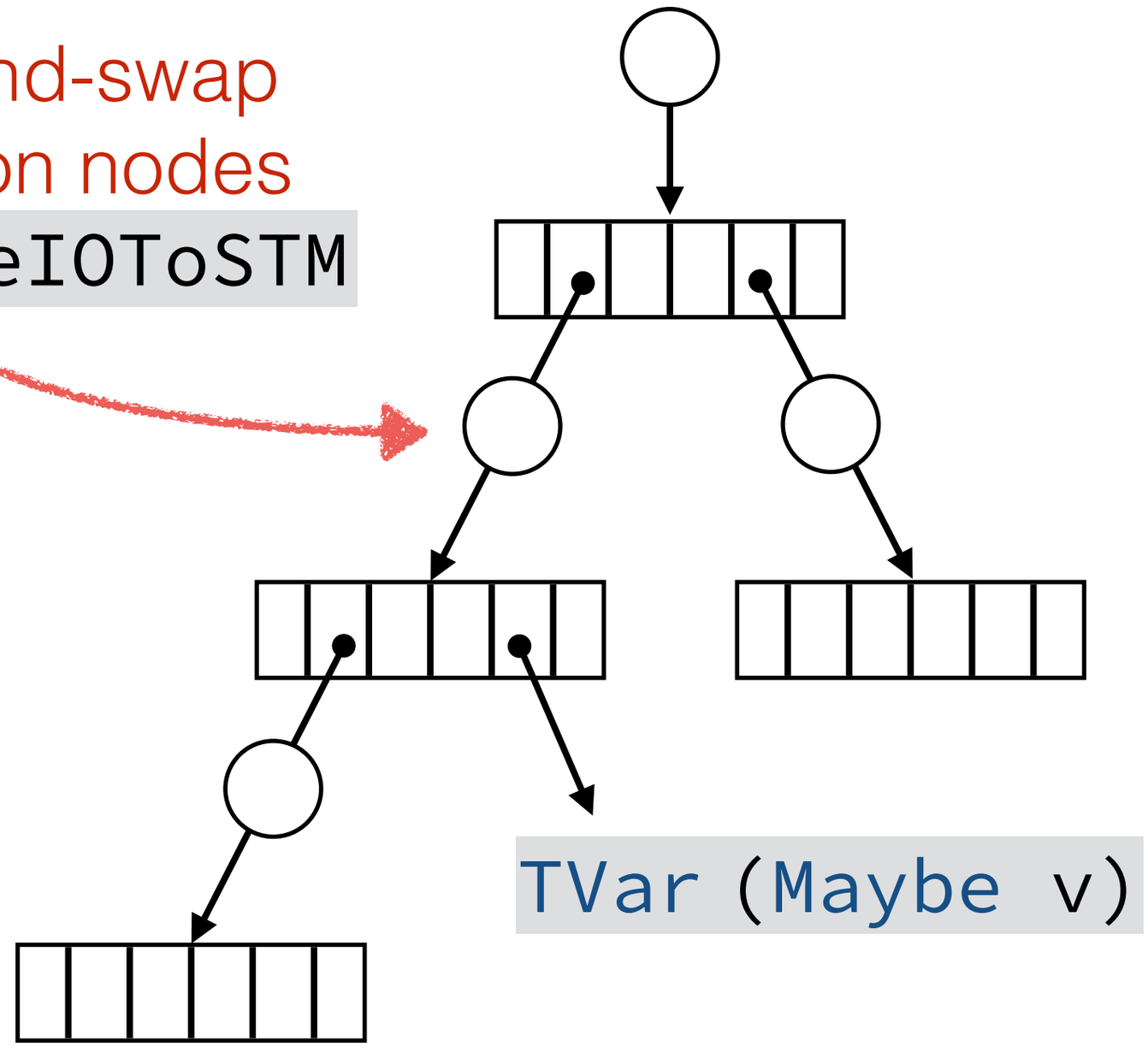
Prokopec, Bagwell, and Odersky

Cache-Aware Lock-Free Concurrent Hash Tries

Tech. Rep. EPFL-REPORT-166908, 2011

Transactional Trie

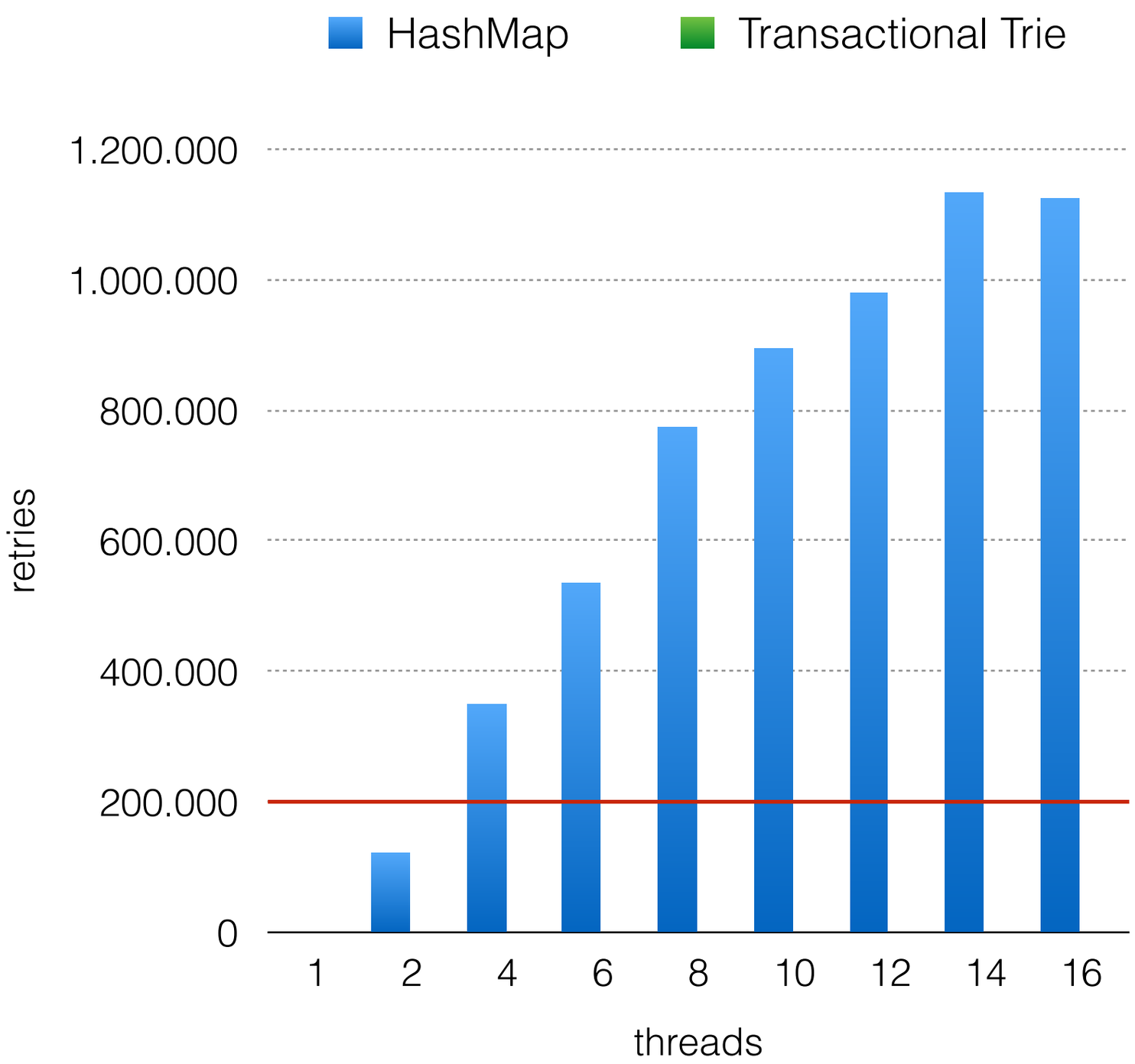
compare-and-swap
on indirection nodes
using `unsafeIOToSTM`

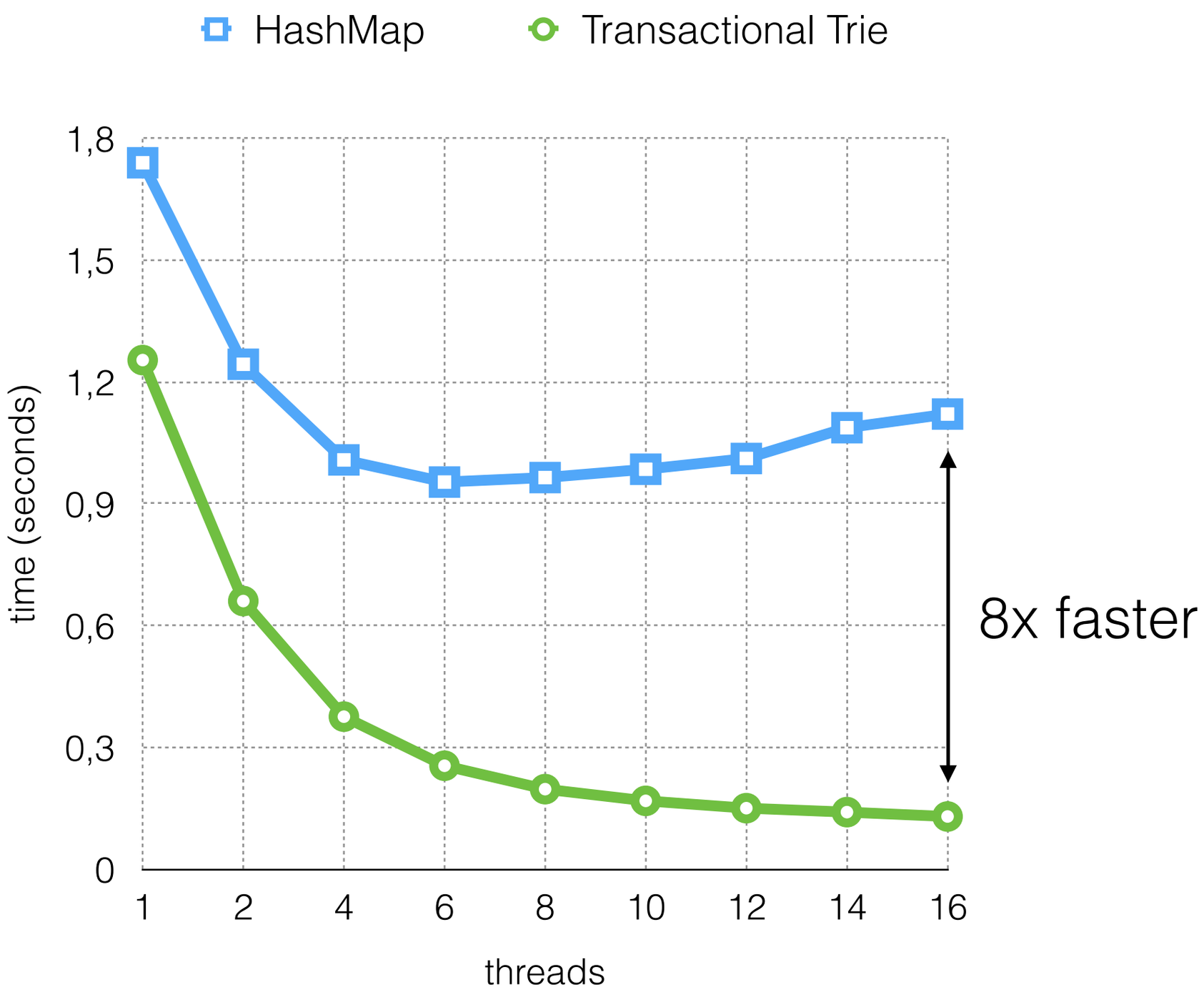


```
getTVar :: k
        → Map k v
        → STM (TVar (Maybe v))
```

- returns the `TVar` stored for `k`
- if there is no `TVar` for `k` yet,
  inserts a new `TVar` with atomic CAS
- $\texttt{getTVar}\ k_1\ m \equiv \texttt{getTVar}\ k_2\ m \iff k_1 \equiv k_2$
- `getTVar` does not read nor write any `TVar`s

```
insert k v m = do var ← getTVar k m
                  writeTVar var (Just v)

delete k m = do var ← getTVar k m
                writeTVar var Nothing

unsafeDelete k m = …

lookup k m = do var ← getTVar k m
                readTVar var

phantomLookup k m = …
```


HashMap
Transactional Trie
1.200.000
1.000.000
800.000
600.000
400.000
200.000
0
retries
1
2
4
6
8
10
12
14
16
threads


HashMap
Transactional Trie
1,8
1,5
1,2
0,9
0,6
0,3
0
time (seconds)
1
2
4
6
8
10
12
14
16
threads
8x faster

github.com/mcschroeder

```
cabal install ttrie
```