# Algorithmic Improvements for Fast Concurrent Cuckoo Hashing

Xiaozhou Li (Princeton), David G. Andersen (CMU), Michael Kaminsky (Intel Labs), Michael J. Freedman (Princeton)

## I. Goal: Fast Concurrent Read & Write

A. Memory efficient (e.g., 95% space utilized)

B. Fast concurrent reads

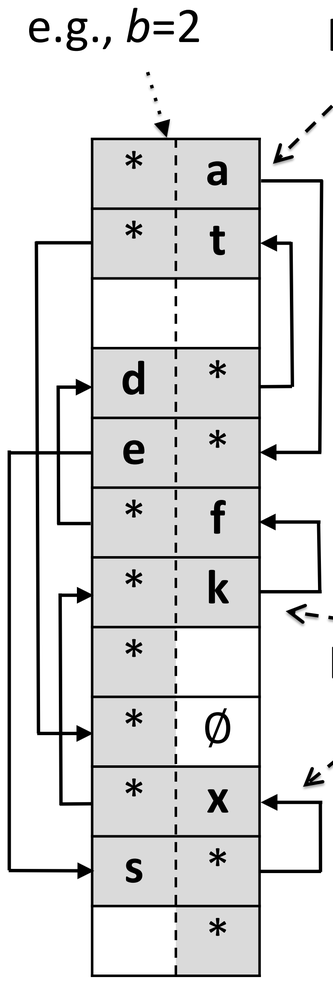
C. **Fast concurrent writes** (scale with # of cores)

## II. Design Principles

- Minimize critical sections
- Exploit data locality
- Optimize concurrency control implementation (e.g., use Intel TSX, hardware transactional memory)

## III. Starting Point: Optimistic Cuckoo Hashing

**MemC3** (*NSDI '13*): goals A and B, but NOT **C**
Each key is mapped to two random buckets
Each bucket has *b* "slots" for items




*Write to an empty slot in one of the two buckets. If all are full, move existing items to their alternate buckets* -> **slow**

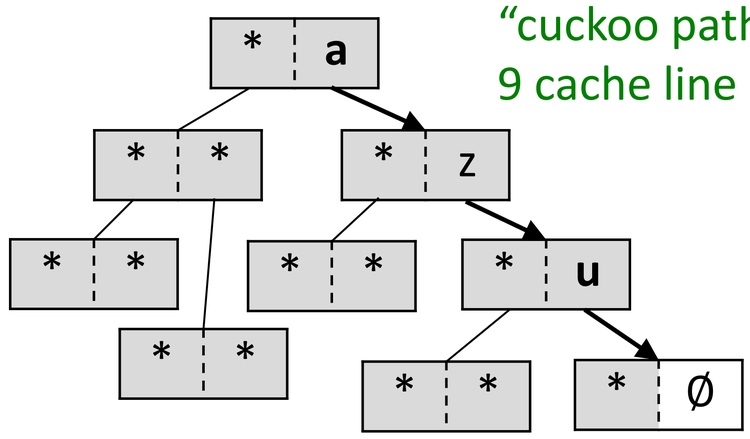
"cuckoo path":
a→e→s→x→k→f→d→t→Ø
9 cache line reads and **9 writes**

*Lock the hash table* -> **single-writer**

*Read two buckets and compare with each slot* -> **fast**
*Use key version counters, no locking for reads* -> **multi-reader**

## IV. Optimizations for `Insert`

- **Breadth-first search for an empty slot**
  - fewer items displaced (logarithmic)
  - enables prefetching



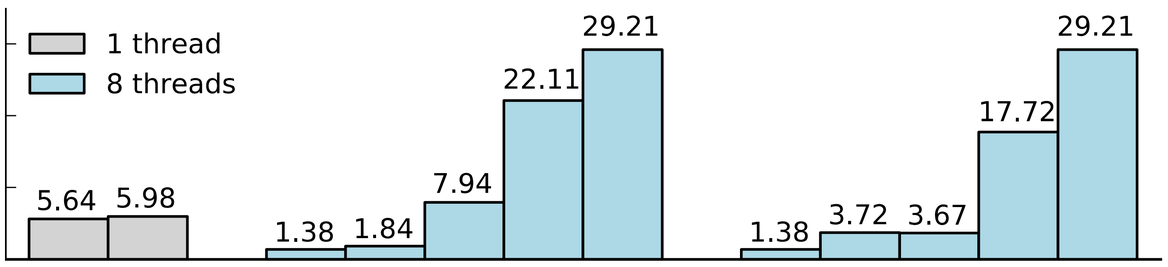

"cuckoo path": a→z→u→Ø
9 cache line reads and **4 writes**

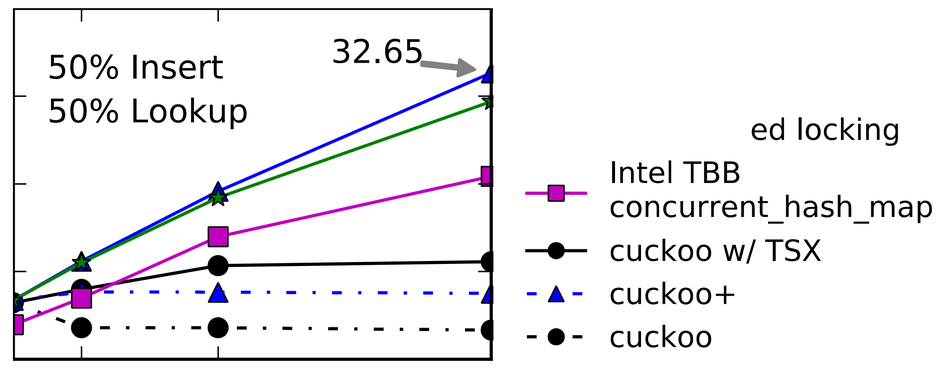
- **Lock *after* discovering an empty slot**
  - minimize the length of critical section
- **Increase set-associativity**
  - fewer items displaced
  - fewer random (more sequential) memory reads
- **Improve concurrency control**
  - use global locking and optimized TSX lock elision
  - *or* use fine-grained spinlock and lock-striping

## V. Evaluation (2 GB hash table, ~134.2 M entries, 8 byte keys and 8 byte values)

Platform: Intel Haswell i7-4770 @ 3.4GHz, 4 cores (8 hyper-threaded cores), 16 GB DRAM, 8 MB L3-cache

Fill a cuckoo hash table from empty to 95% capacity




### 100% Insert throughput with Intel TSX

TSX-glibc: Intel library for TSX lock elision
TSX*: our optimized implementation (detailed in paper)

### Throughput vs. # of threads

cuckoo: single-writer/multi-reader hashing in MemC3.
cuckoo+: our optimized cuckoo hashing

Intel Science & Technology Center for Cloud Computing

*In Proc. EuroSys'14*
*github.com/efficient/libcuckoo*

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley · University of Washington