

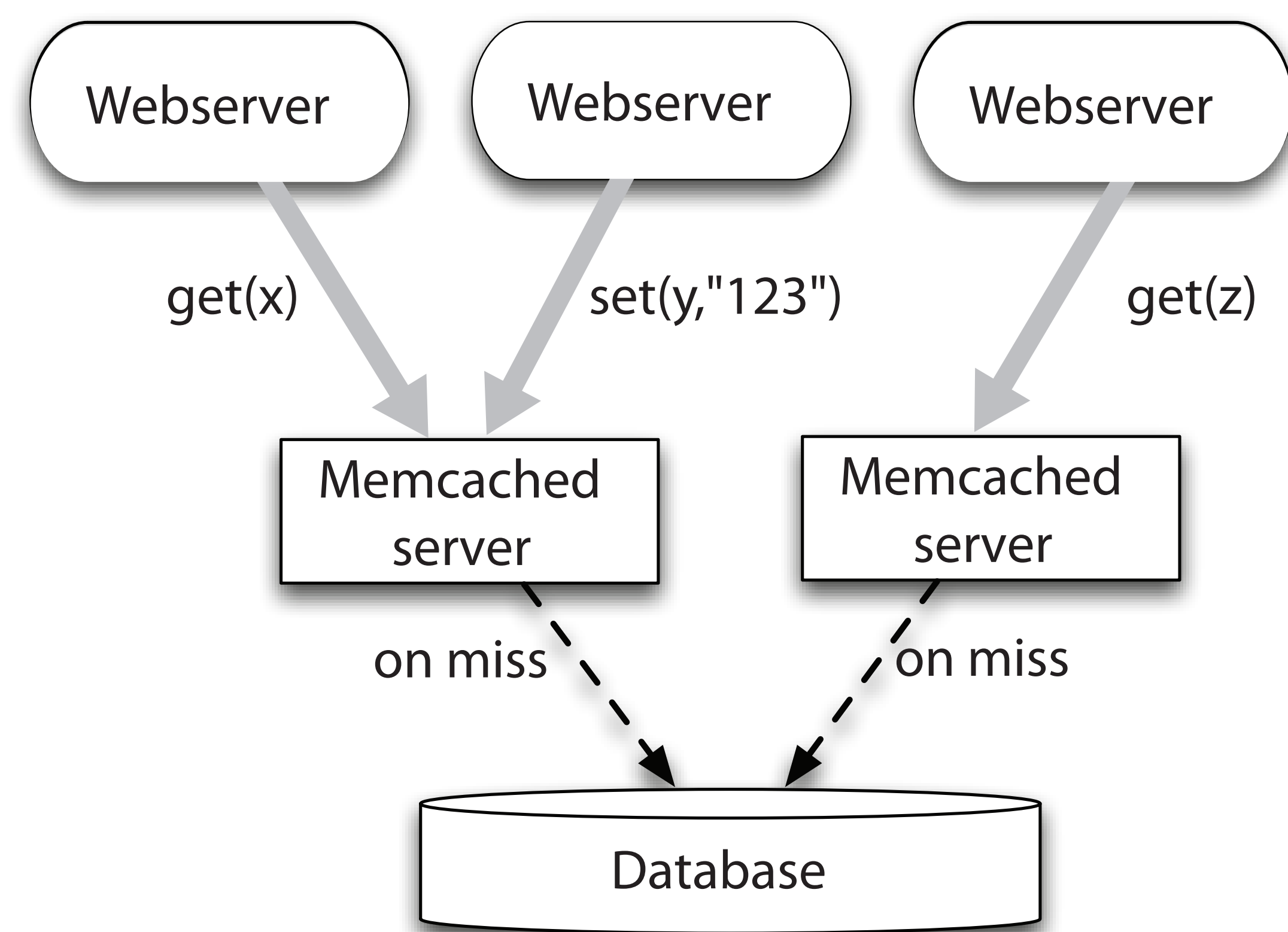
MEMC3: MEMCACHE W/ CLOCK AND CONCURRENT CUCKOO HASHING

Bin Fan, David G. Andersen (Carnegie Mellon University), Michael Kaminsky (Intel Labs Pittsburgh)

MEMCACHED BACKGROUND

DRAM-based object cache

- Speed up web applications
- Alleviate database load.



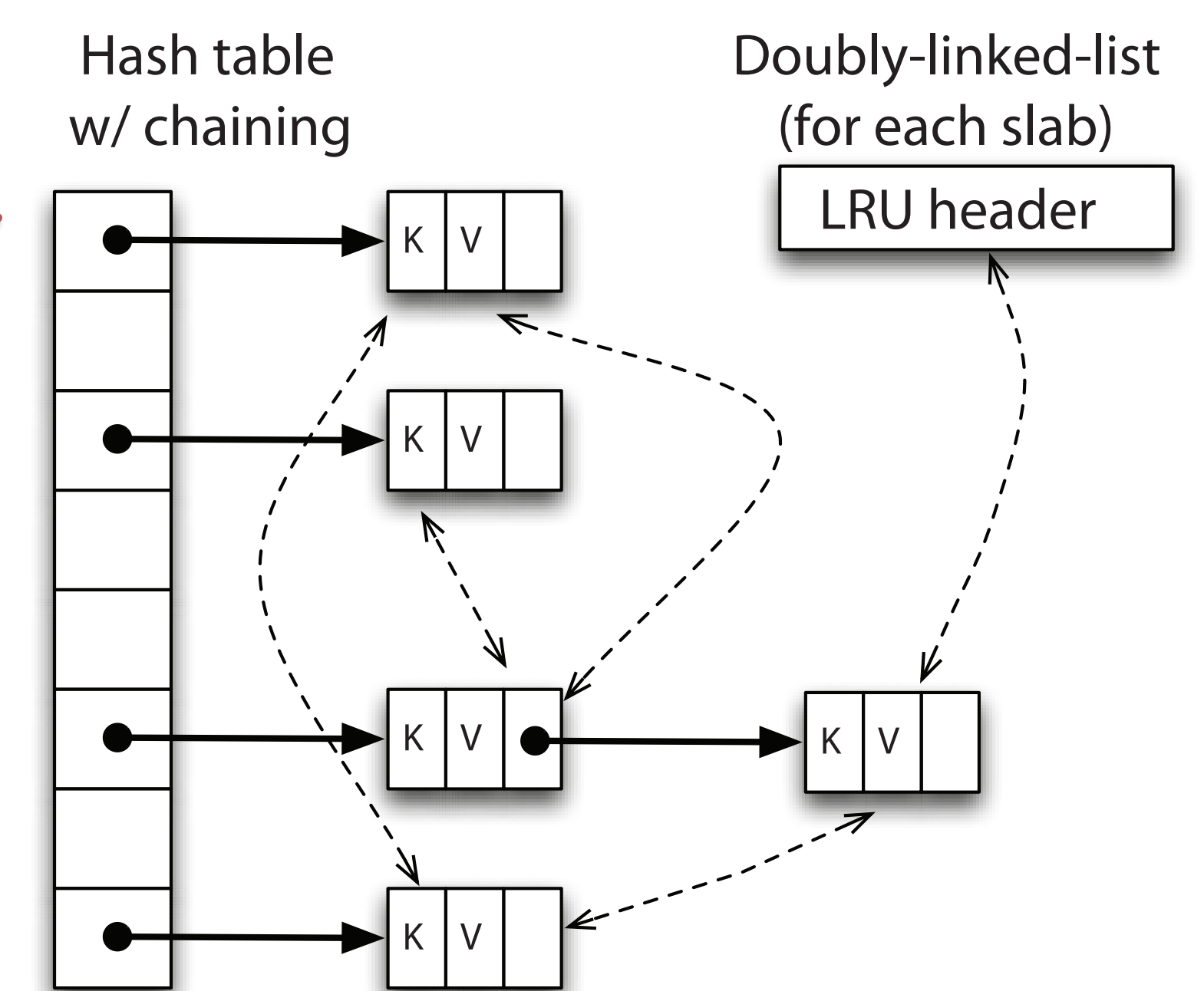
CORE DATA STRUCTURES

Perf
Poor multi-core scalability

- serialized hashtable and LRU operations
- tput doesn't scale

Mem
High space overhead

- often used for small objects [1]
- 100B object + 56B header => overhead > 50%



OPTIMIZATIONS

COMPACT DATA STRUCTURES **Perf** **Mem**

- Replace chaining with "optimistic cuckoo hashing" [2]
 - Max hash table occupancy → 93%
 - Support single-writer/multi-reader access
- Change linked list-based LRU to CLOCK
 - Replace two pointers with a reference bit for each object

ARCHITECTURE-AWARE OPTIMIZATIONS **Perf**

- CPU cache locality
- Instruction/memory level parallelism

WORKLOAD-AWARE OPTIMIZATIONS **Perf** **Mem**

- Read mostly (e.g., > 98% [1])
- Small objects dominate

SYSTEM TUNING **Mem**

- Pin thread on one core
- Increase page size: fewer TLB misses
- Purpose-built memcmp for small keys: compare 32bit at a time

CONCLUSIONS

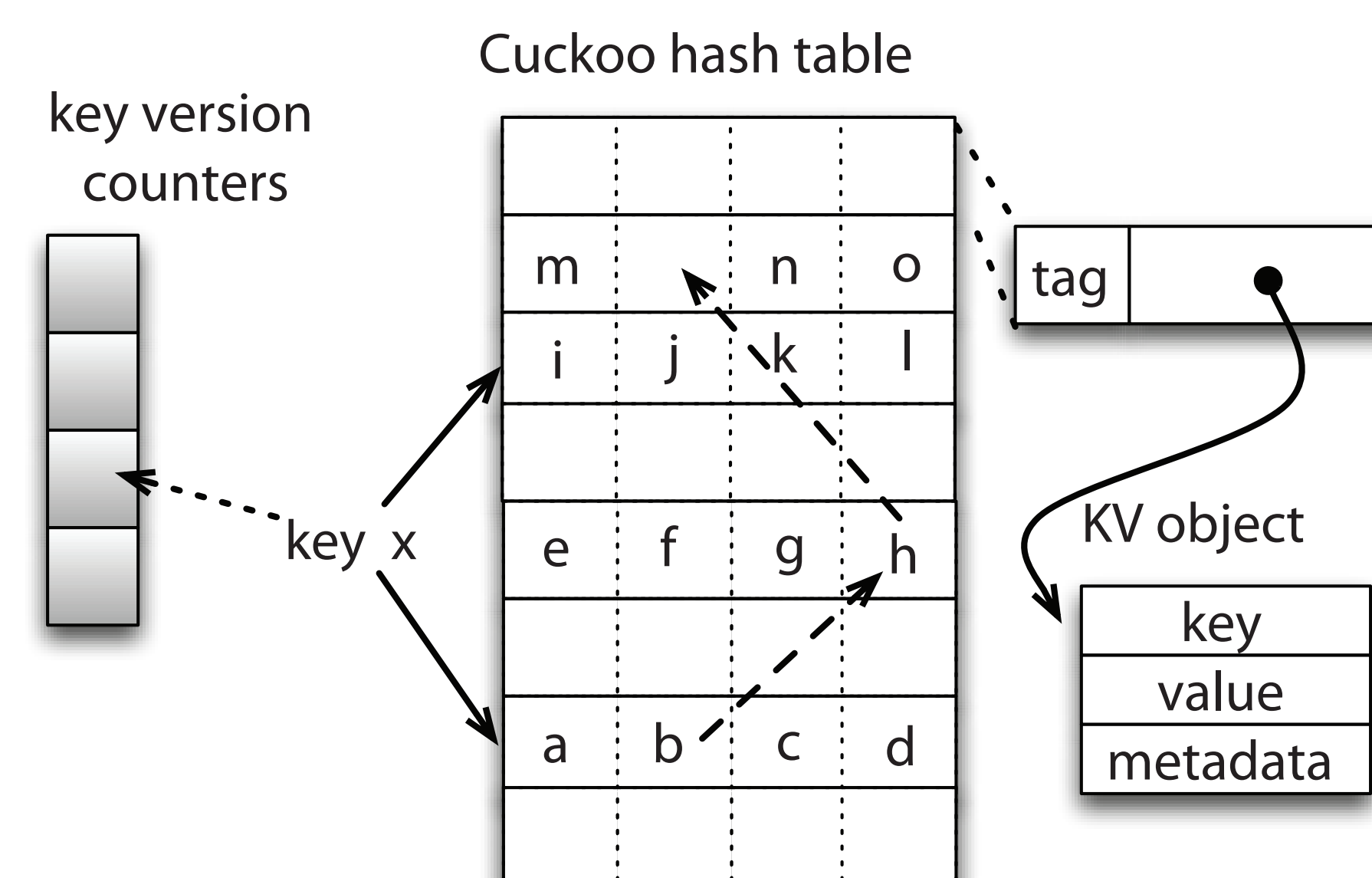
- Memcached is bottlenecked by its core data structures when scaling to multiple cores
- Compact and concurrent algorithms improve significantly memcached throughput and space efficiency.

[1] Atikoglu et al., "Workload analysis of a large-scale key-value store", SIGMETRICS2012

[2] Lim et al., "SILT: A memory-efficient, high-performance key-value store", SOSP2011

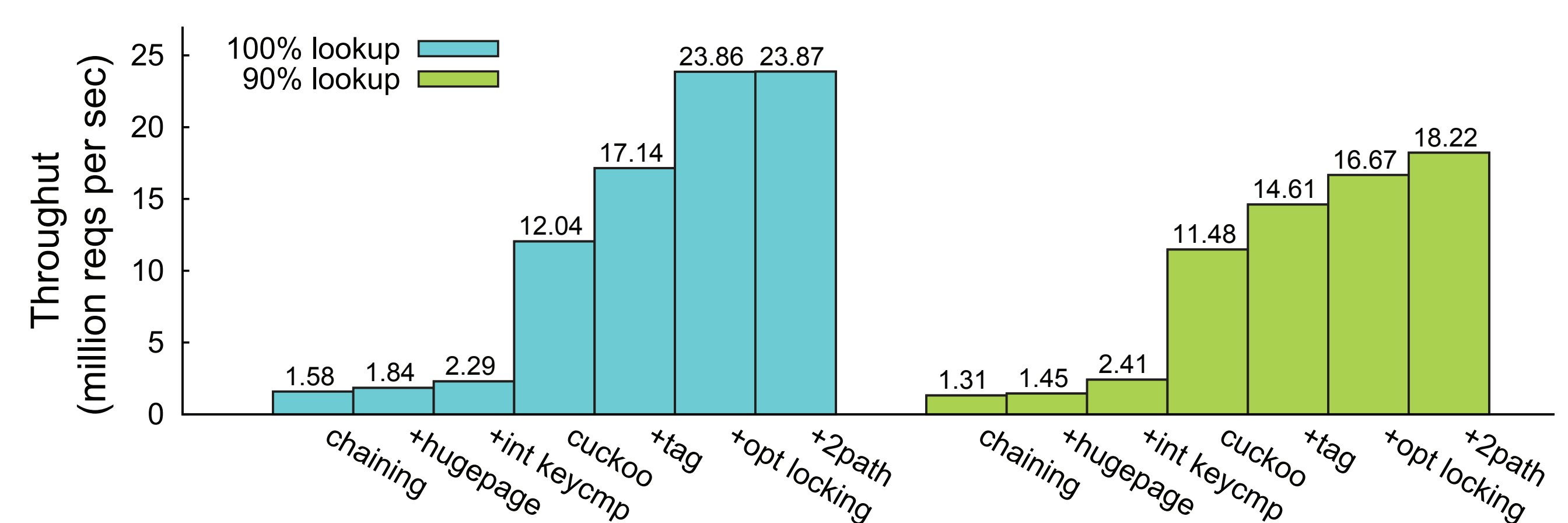
OPTIMISTIC CUCKOO HASH TABLE

- Map each key to two buckets
- Lookup: 3 memory reads
- Insert: O(1) w/ recursive displacement
- Increment key version before/after each displacement
- Compare key version snapshots before/after read



EVALUATION

HASHTABLE PERFORMANCE



- 6 threads accessing a hashtable (~1GB)
- Results are independent of key-value size

END-TO-END PERFORMANCE

- 16 B key + 32 B value, 95% get + 5% set
- 50 remote clients, one 10Gb NIC on server
 - Store 25% more objects
 - Scales to 12 cores with 4.3 MOPS or ~3x improvement

