Lock Free RW Concurrency in hash library

HONNAPPA NAGARAHALLI
ARM

Agenda

DPDK
DATA PLANE DEVELOPMENT KIT

- Current Issues
- Ingredients for lock free algorithms
- Design for lock free RW concurrency in rte_hash
- Reclaiming Memory
- Plan for upstreaming

Current Issues

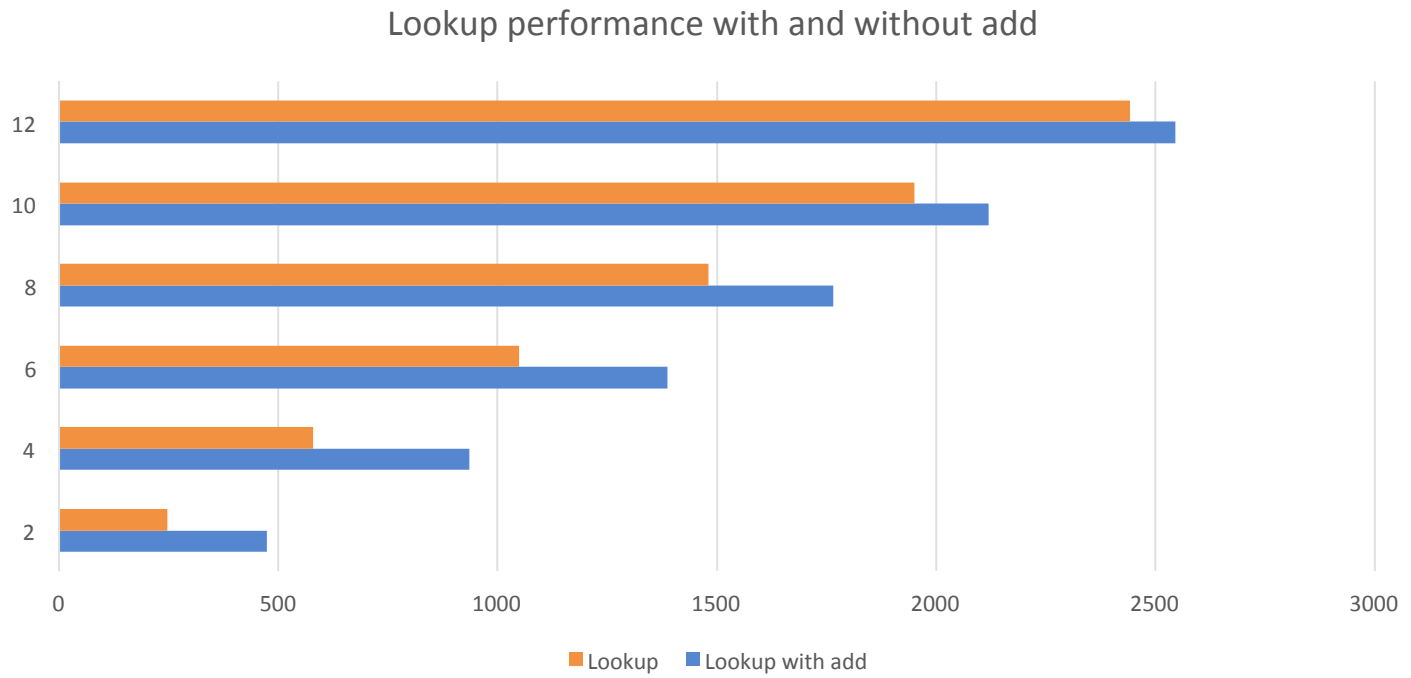
- Preempted writer will block the readers
  - Problem applies for platforms with HTM when it falls back to traditional locks

```
static inline void rte_rwlock_write_lock_tm(rte_rwlock_t *rwl)
{
    if (likely(rte_try_tm(&rwl->cnt)))
        return;
    rte_rwlock_write_lock(rwl); == fallback to traditional lock
}
```

- Application uses the key store index to reference its data
  - Index should not be freed till the application has stopped using it

Current Issues

- Performance of lookups in the presence of writers



Ingredients for lock free algorithms

DPDK
DATA PLANE DEVELOPMENT KIT

- Atomic operations – sizes matter, look for wider support
- Memory ordering operations – Memory barriers or C11 atomics
  - Orderings by themselves are not enough. Need to identify the ‘payload’ and ‘guard’
  - Payload – the data being propagated from writer to reader, accesses are not required to be atomic
  - Guard – Protects access to the payload, accesses need to be atomic
  - Synchronizes with relationship between writer and reader

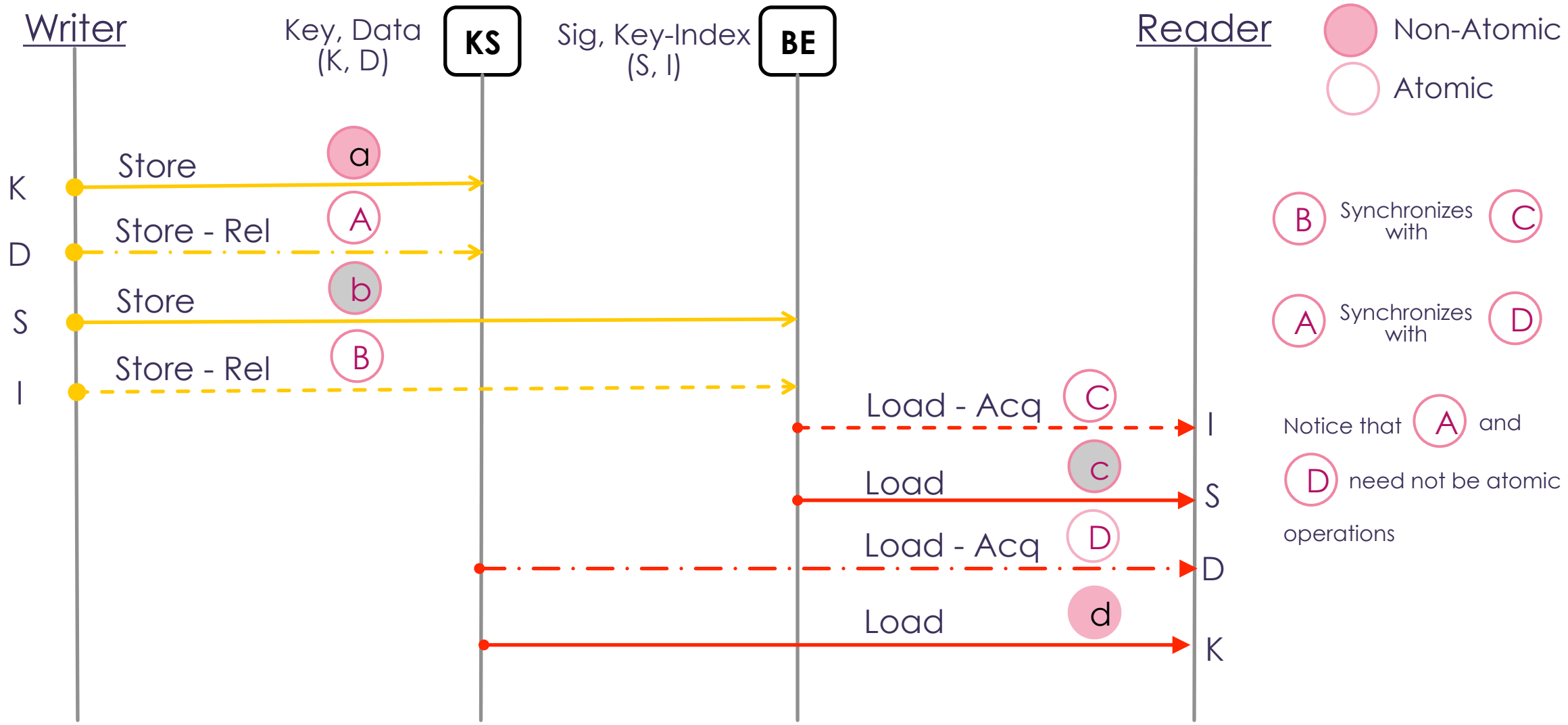
Ingredients for lock free algorithms

- Data structure specific challenges
- Re-claiming memory
  - Readers continue to reference an entry in the data structure even the after delete
  - Memory cannot be ‘freed’ immediately after ‘delete’
  - Delete – Remove the reference to memory/entry
  - Free – Returning the memory/entries to free pool
  - Mechanisms are required to identify when to ‘free’ the entry/memory

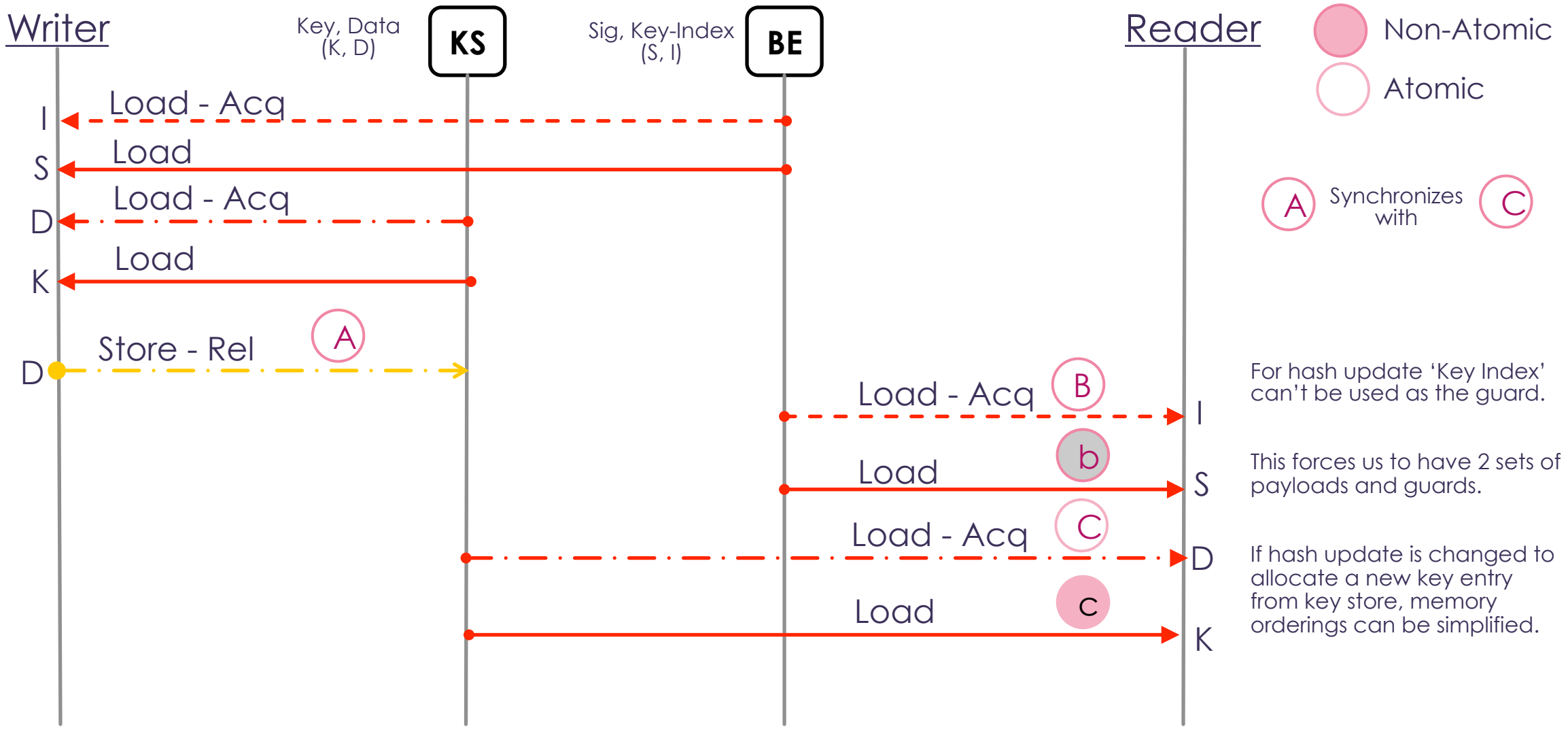
Design

- Atomic Operations – relying on 32b operations
- Memory orderings – working with C11 atomic functions
  - Payload 1 – key (stored in key store)
  - Guard 1 – pdata (stored in key store)
  - Payload 2 - current signature and alternate signature (stored in bucket entry)
  - Guard 2 – Index to {key, pdata} entry in the key store (stored in bucket entry)

Design – Ordering Mem Operations – Hash Add/Lookup

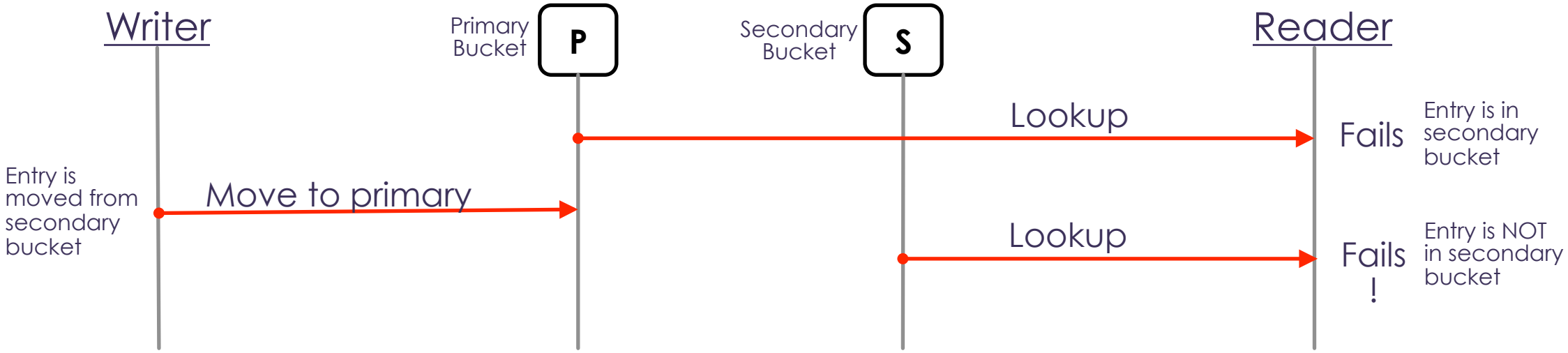


Design – Ordering Mem Operations – Hash Update/Lookup



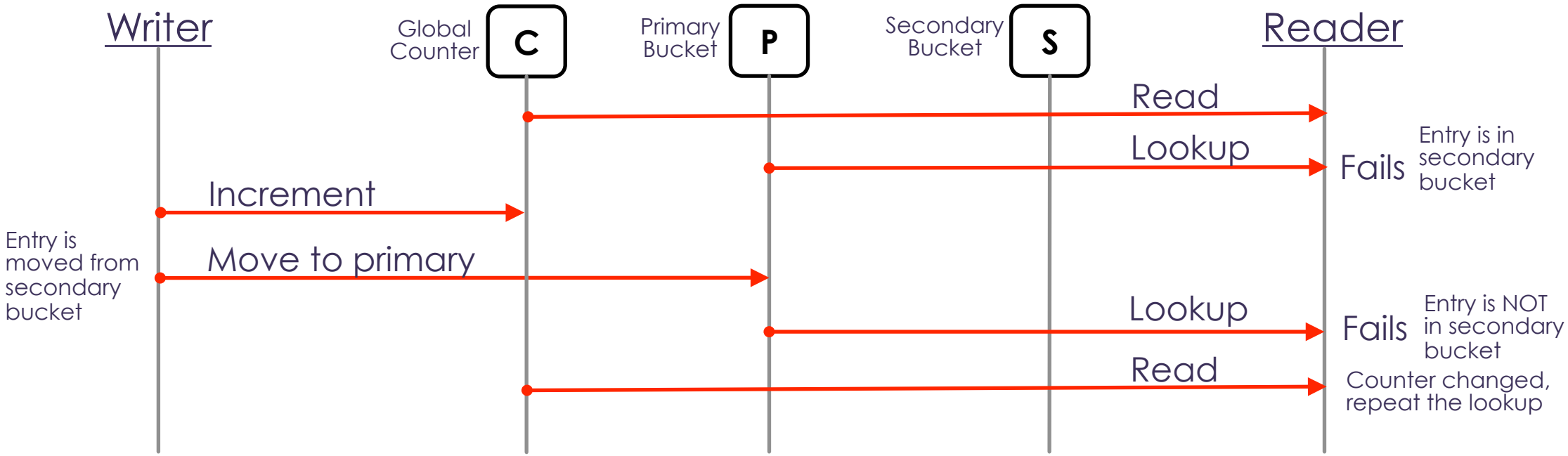
Design – Data Structure Specific Challenge

- Hash add can move entries to their alternate positions
- Due to concurrent adds, reader might not find the entry even though it is present



Design – Data Structure Specific Challenge

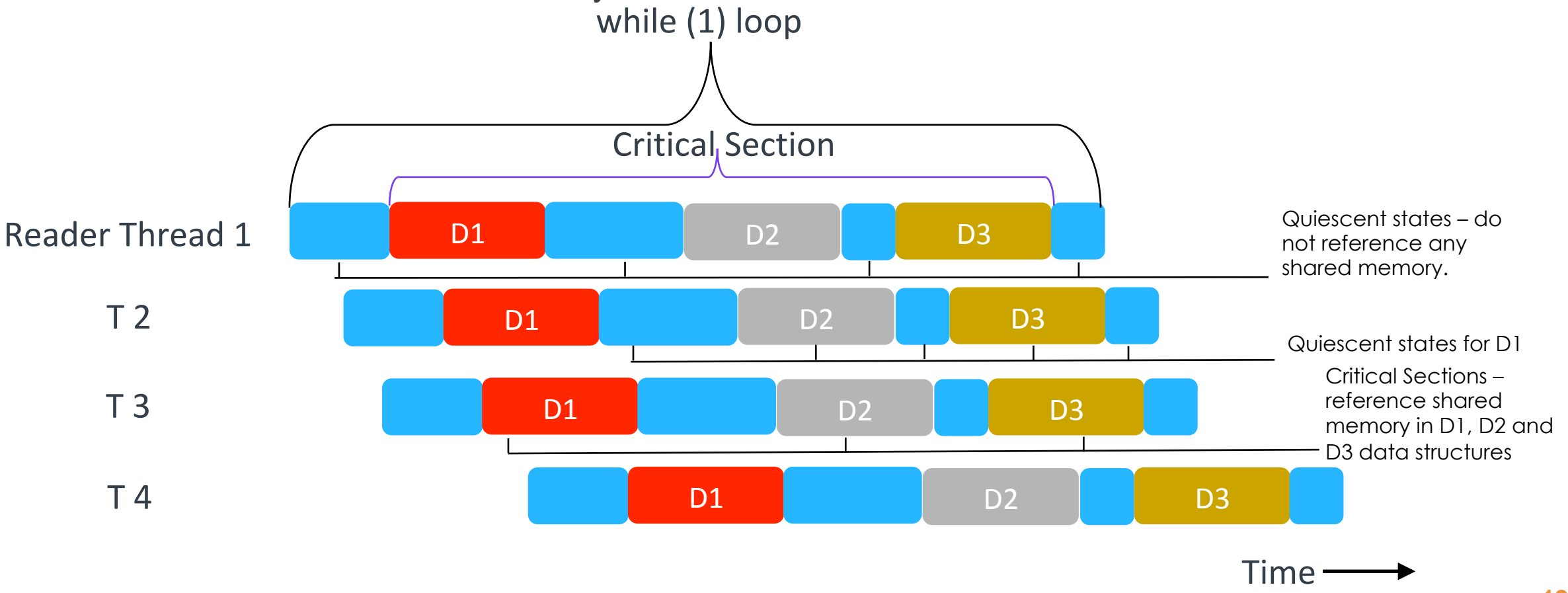
- Solution uses a global counter
- Counter indicates to the reader that the table has changed



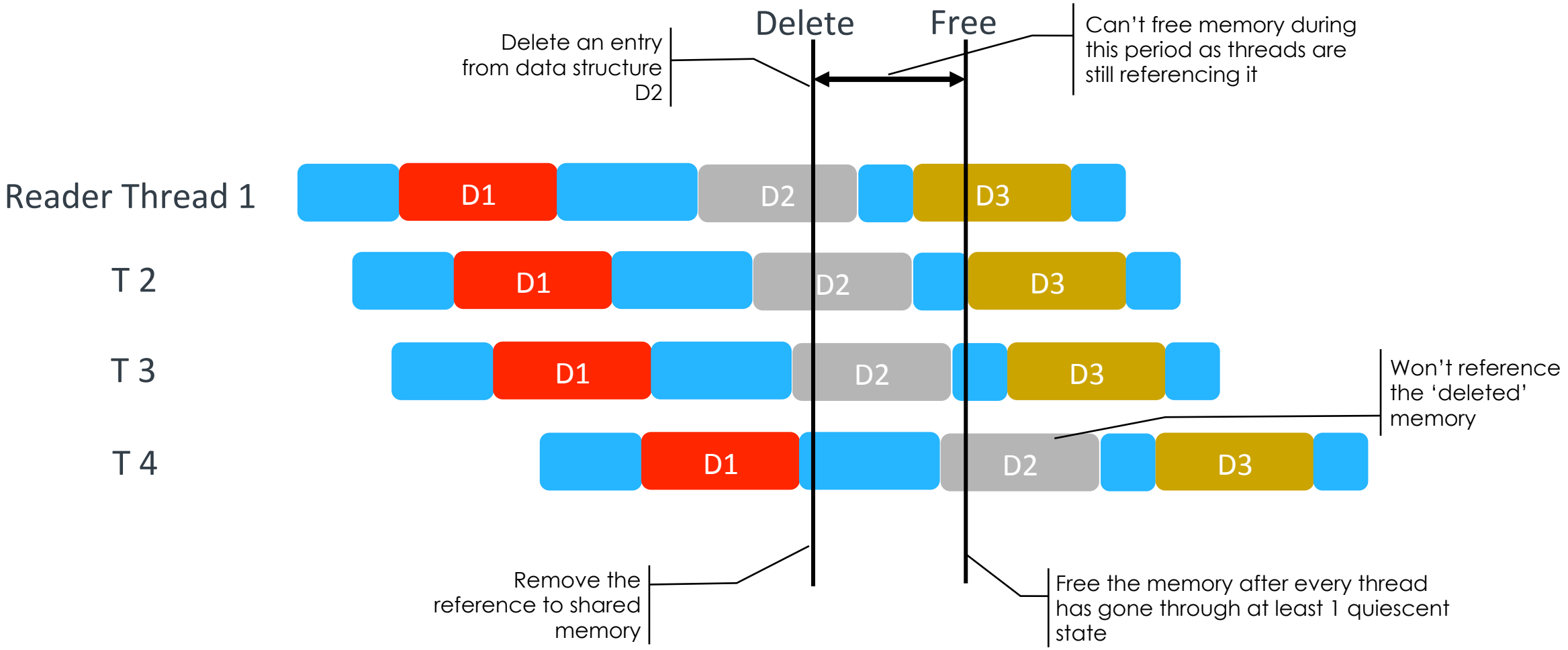
- Entry present but not moving – entry is found immediately
- Entry present but moving – reader has to chase it
- Entry not present – Repeats till the move stops – Can be improved by using bucket counter

Re-claiming Memory – TQS

- Thread Quiescent State (TQS) - Any place in the code where the thread does not hold a reference to shared memory



Re-claiming Memory – Delete/Free



Re-claiming Memory – rte_tqs library

- rte_tqs library
  - Provides the ability to check if a set of reader threads have entered at least 1 quiescent state
- Goals
  - Provide flexibility to check the quiescent state
    - Single data structure, a group of data structures or any application defined granularity
  - Ability to check the quiescent state of a given set/all of readers
  - Ability to check quiescent state synchronously
  - Ability to check quiescent state at a later point

Re-claiming Memory – rte_tqs library

- APIs
  - rte_tqs_create – creates a TQS variable
  - rte_tqs_start – triggers the readers to inform the writer about completion of ‘n’ number of quiescent states
  - rte_tqs_get – checks if the threads have passed through ‘n’ number of quiescent states
  - rte_tqs_update – called by the data plane threads to update the tqs state
  - rte_tqs_delete – free tqs checker entry

Plans for upstreaming

- Lock Free RW concurrency patch will be sent to mailing list for review (beginning of next week) – Targeting 18.11
- TQS library – High Level Design, API definition - Done
  - Coding/Testing need to happen – Targeting Q4 for 1st patch

Questions?

Scaling with Lock Free RW concurrency

- Numbers with patch