Extending DPDK Flow Classification Libraries for Determinism & Cloud Usages

SAMEH GOBRIEL
INTEL LABS

Contributors

- Yipeng Wang yipeng1.wang@intel.com
- Ren Wang ren.wang@intel.com
- Charlie Tai charlie.tai@intel.com

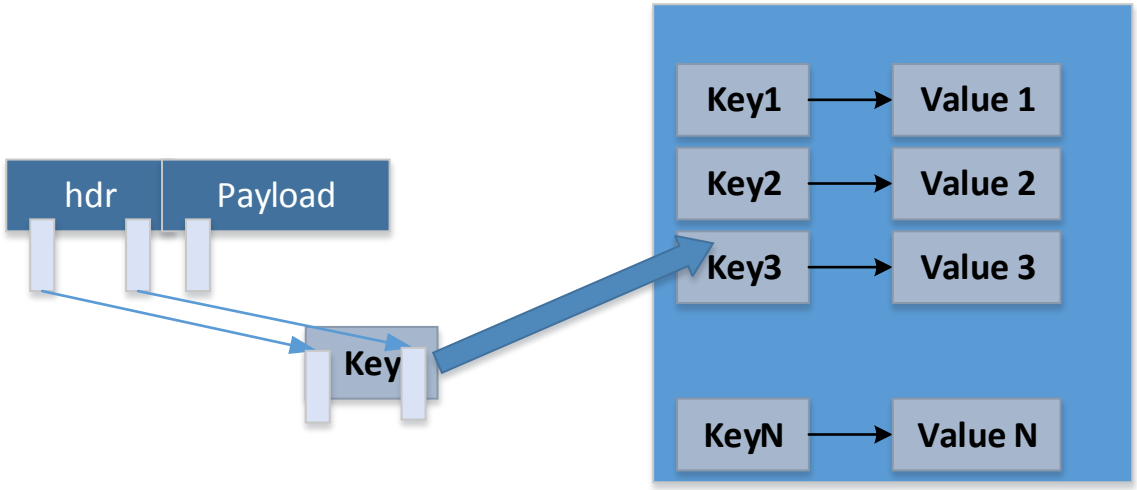
- And thanks for all the inputs from Bruce Richardson, Pablo De Lara Guarch, Cristian Dumitrescu and John Mcnamara.

Agenda

- Overview of Flow Classification Libraries in DPDK
  - Hash Library
  - EFD (Elastic Flow Distributor) Library
  - Membership Library
- Recent optimizations for rte_hash in DPDK v18.08
- Upcoming optimizations for rte_hash targeting DPDK v18.11
- Cloud Workloads and Membership Library Research Direction

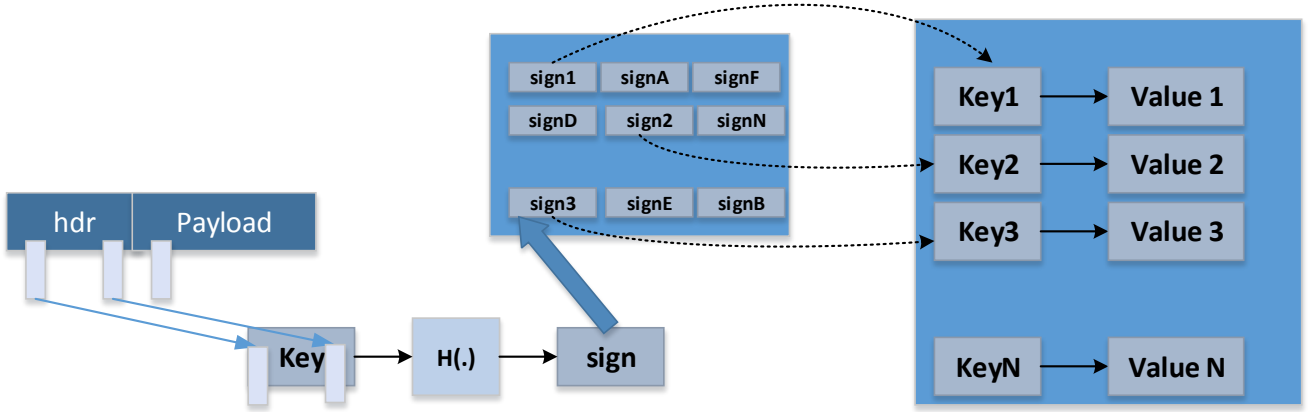
Flow Classification 10,000 ft.

- A flow-table of millions of keys
- For an incoming packet, a lookup key is formed
- The flow-table is looked up and if there is a match, the correct value/ action is retrieved.



Flow Classification 10,000 ft.

- Because of huge number of flows, typically flow-table is optimized for a “miss”.
- To minimize full key comparison cost, flow tables can store signatures (shorter than full key).
- Signatures are calculated by hashing the full key
- When a signature match occurs the full key is compared and if matched the value/action is retrieved.



DPDK Flow Classification Libraries

DPDK
DATA PLANE DEVELOPMENT KIT

Hash Library

Membership Library

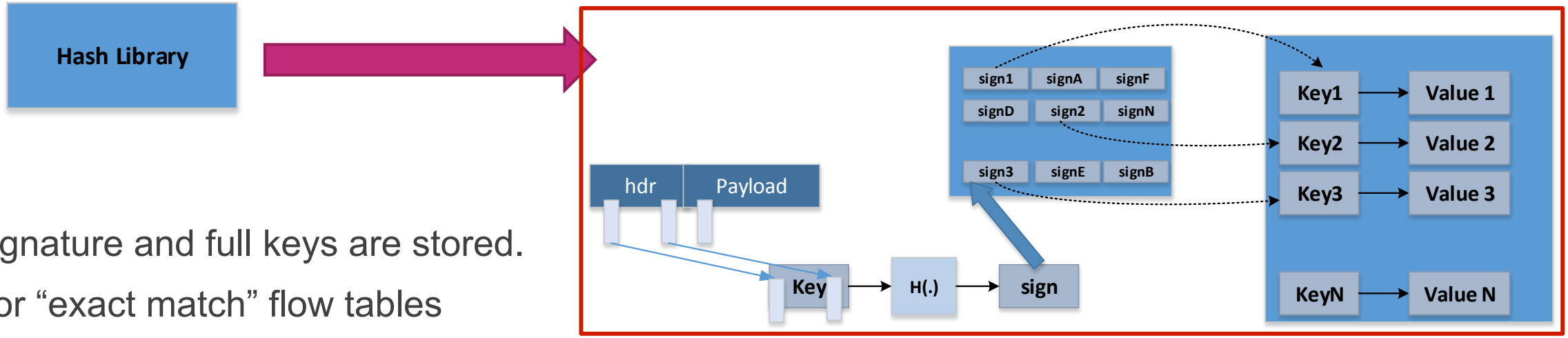
EFD Library

- Each has for their own usages and applications.
- They differ basically in what we store in the flow-table, and hence, whether false positives are allowed
- Each library is designed to provide the best performance for its intended usage.

"**Confusion** is a word we have invented for an order which is not yet understood"

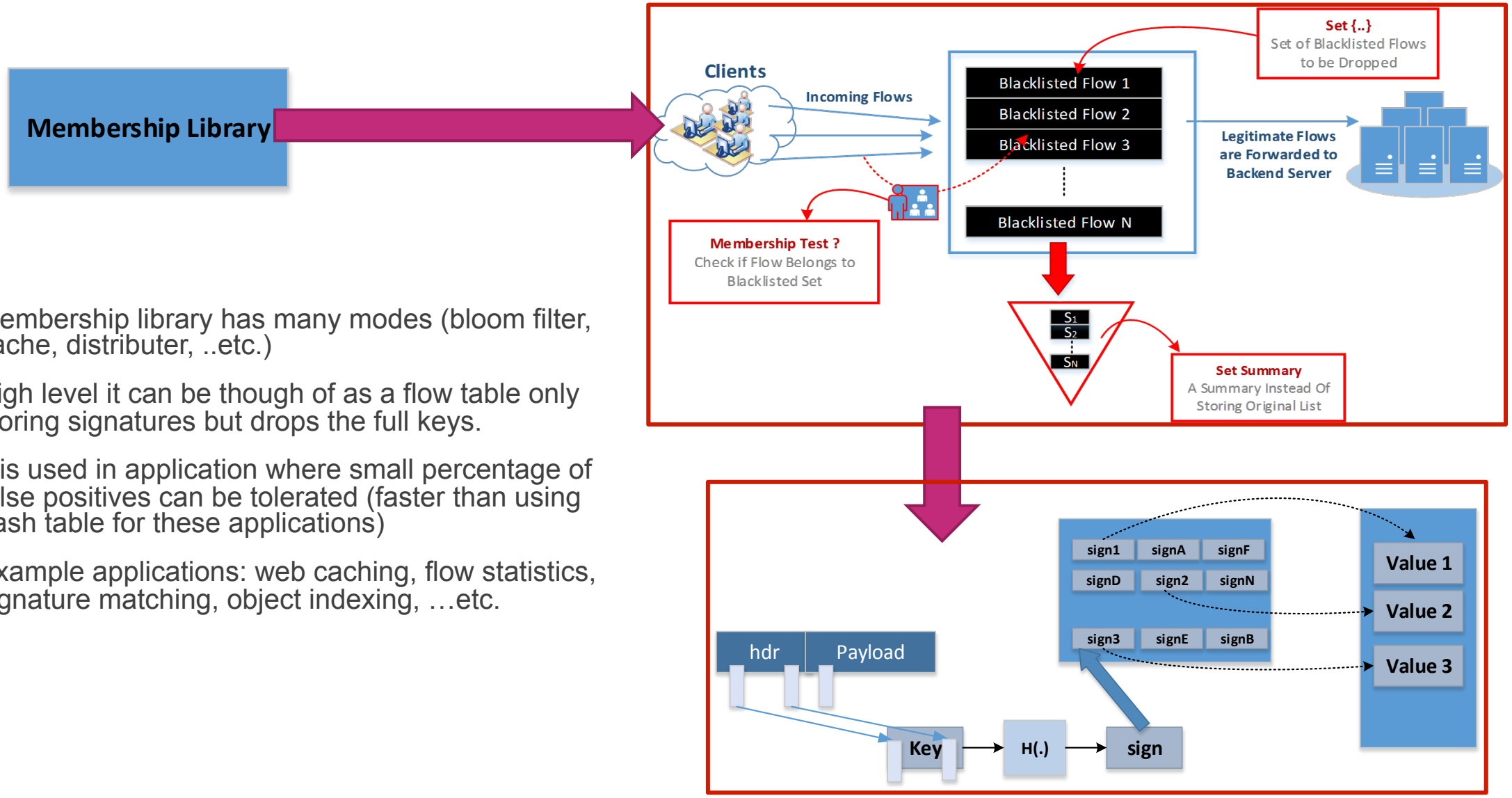
Henry Miller
American Writer
(1891 – 1980)

DPDK Flow Classification Libraries



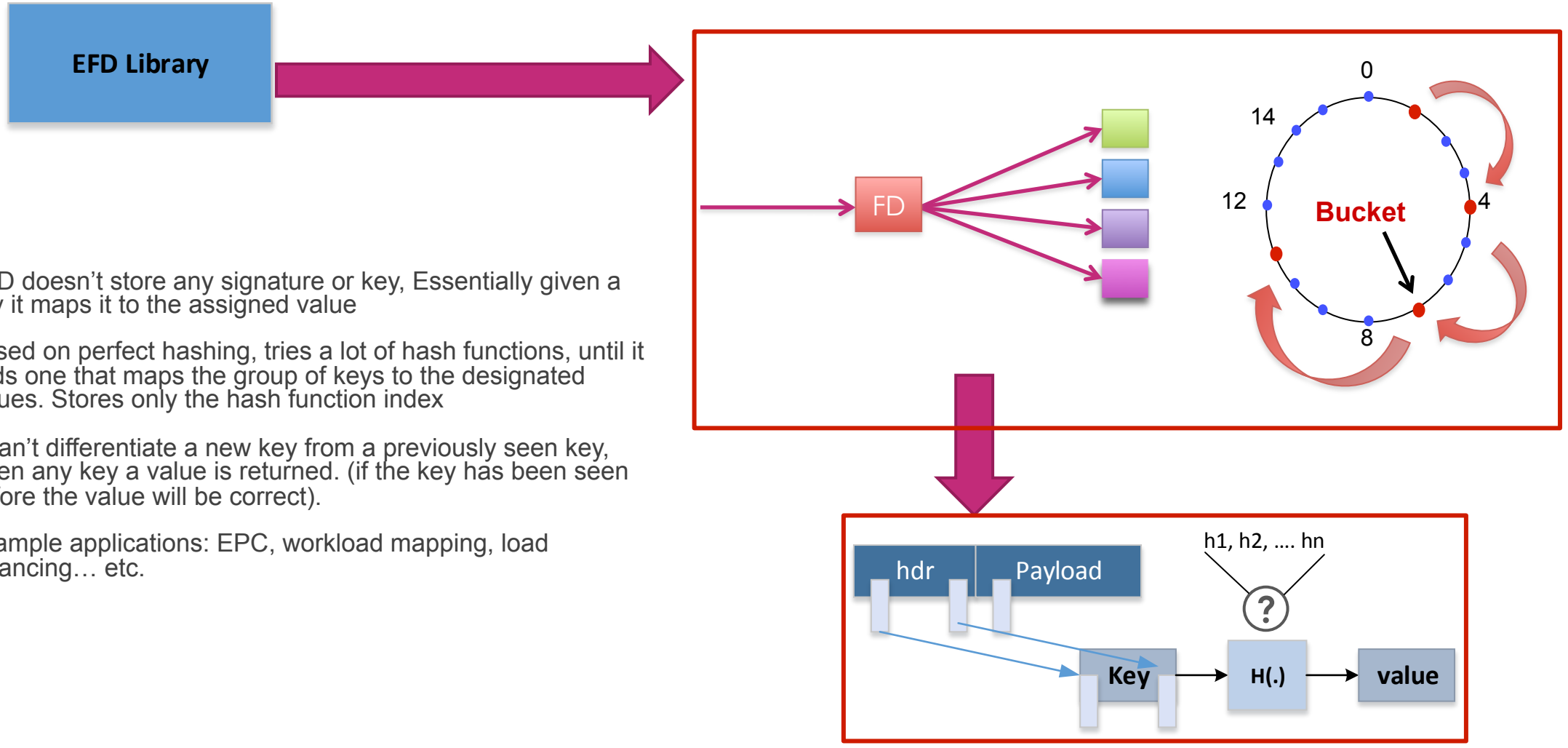
- Both signature and full keys are stored.
- Used for “exact match” flow tables
- No False positive or false negatives (100% sure when we miss that we haven’t seen this key before, and 100% sure when hit that the associated value is assigned to this specific key).

DPDK Flow Classification Libraries



- Membership library has many modes (bloom filter, cache, distributer, ..etc.)
- High level it can be though of as a flow table only storing signatures but drops the full keys.
- It is used in application where small percentage of false positives can be tolerated (faster than using hash table for these applications)
- Example applications: web caching, flow statistics, signature matching, object indexing, …etc.

DPDK Flow Classification Libraries

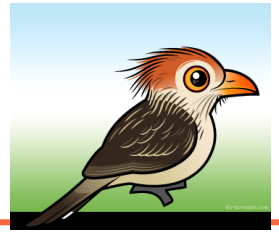


- EFD doesn't store any signature or key, Essentially given a key it maps it to the assigned value
- Based on perfect hashing, tries a lot of hash functions, until it finds one that maps the group of keys to the designated values. Stores only the hash function index
- It can't differentiate a new key from a previously seen key, given any key a value is returned. (if the key has been seen before the value will be correct).
- Example applications: EPC, workload mapping, load balancing… etc.

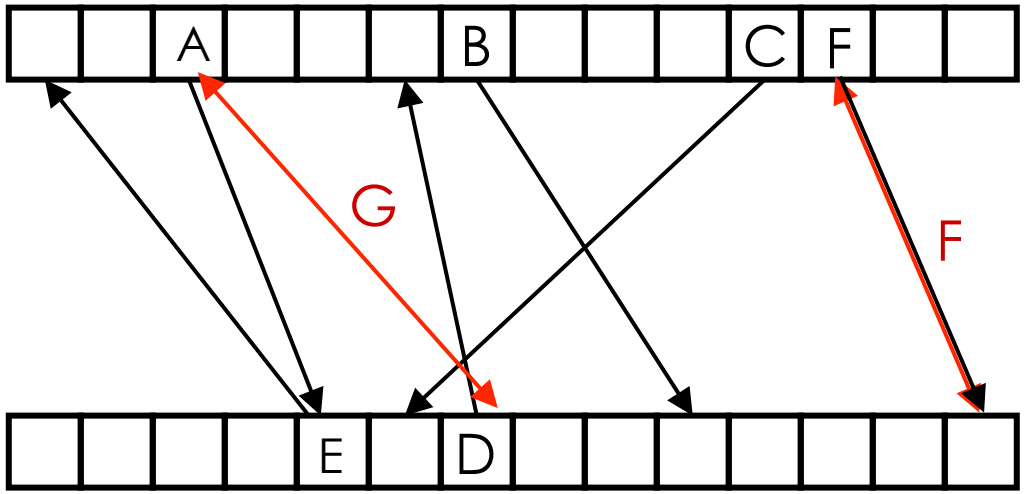
Agenda

- ✓ Overview of Flow Classification Libraries in DPDK
  - ✓ Hash Library
  - ✓ EFD (Elastic Flow Distributor) Library
  - ✓ Membership Library
- Recent optimizations for rte_hash in DPDK v18.08
- Upcoming optimizations for rte_hash targeting DPDK v18.11
- Cloud Workloads and Membership Library Research Direction

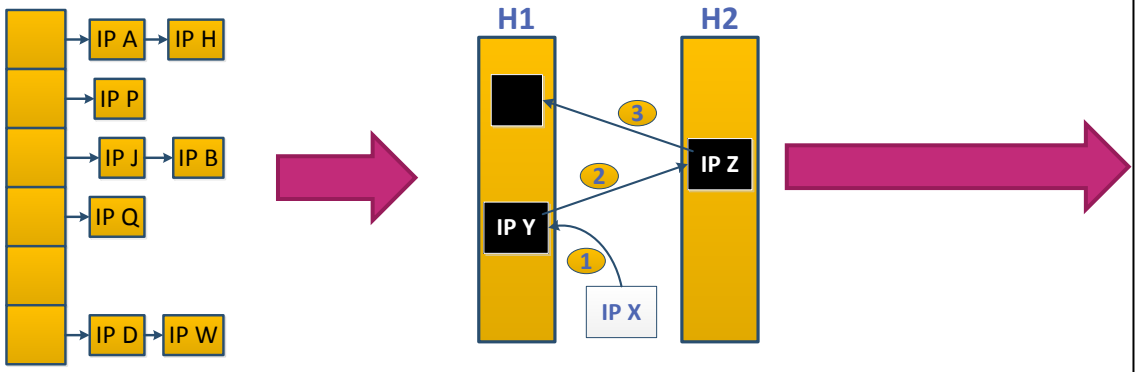
DPDK RTE Hash Library

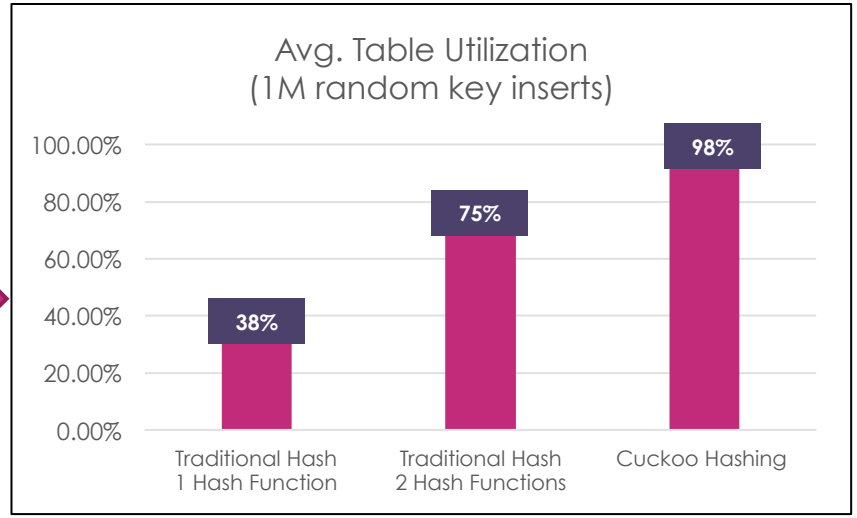


- Library is based on Cuckoo Hashing



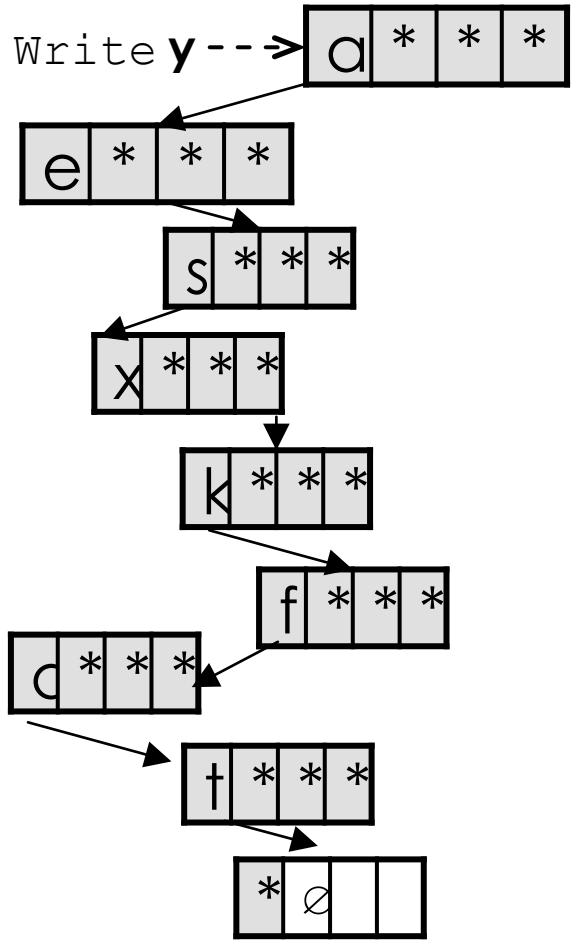
- Provides very good table utilization
  - Denser tables fit in cache.
  - Can scale to millions of entries.
  - Significant throughput improvement





RTE Hash V18.08 with Read/Write Concurrency

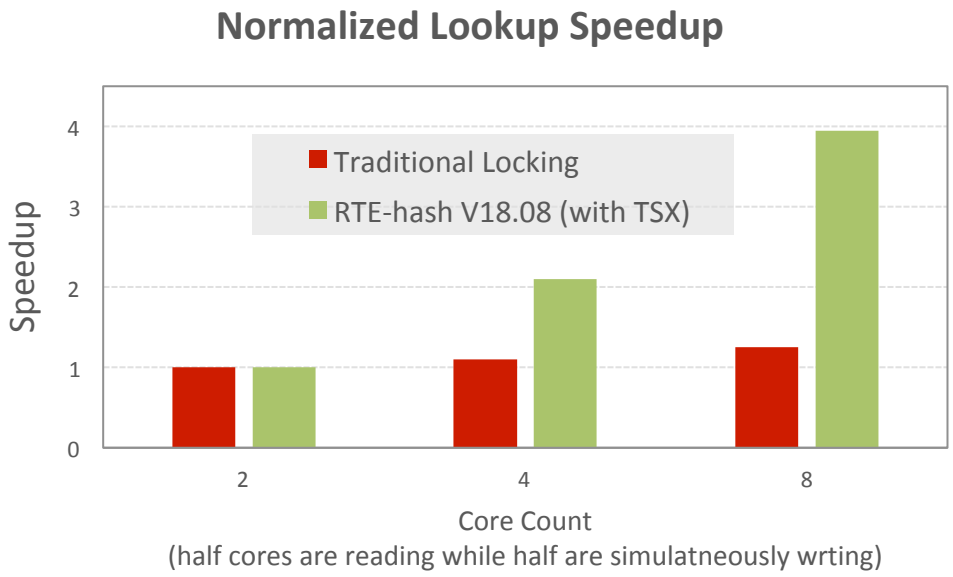
- Especially at high table load, one insert, can cause many keys to get displaced.
- Cuckoo insert path can get very long.
- In earlier DPDK versions, because the writers are changing the content of the buckets, no concurrent readers can go through until writers finish updating
- No R/W Concurrency → Slow down Lookups operations.



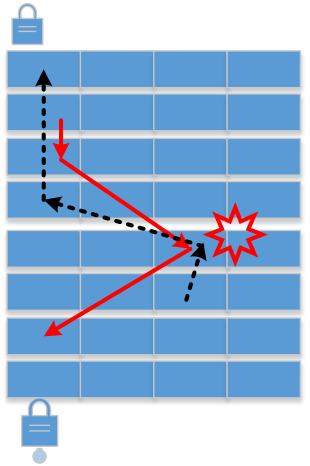
Cuckoo Insert Path:
a→e→s→x→k→f→d→t→∅

RTE Hash V18.08 with Read/Write Concurrency

- Using Intel TSX, the h/w monitors every cache line and if a reader is reading an updated cache line it rolls back execution.

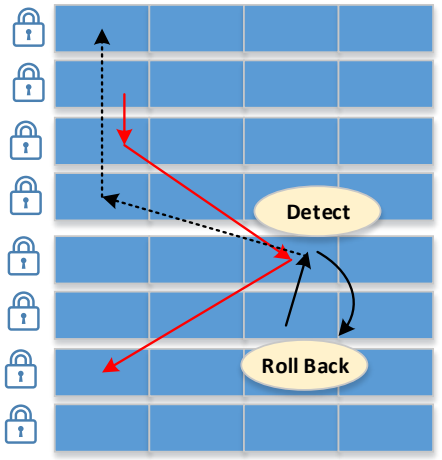


Coarse Grained Locks



- Limited Concurrency
- Threads are serialized in critical section

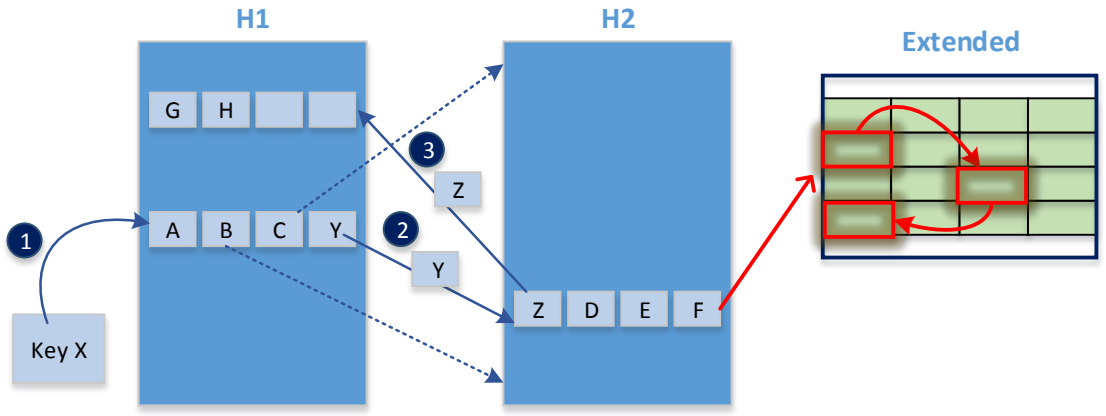
TSX Hardware Concurrency

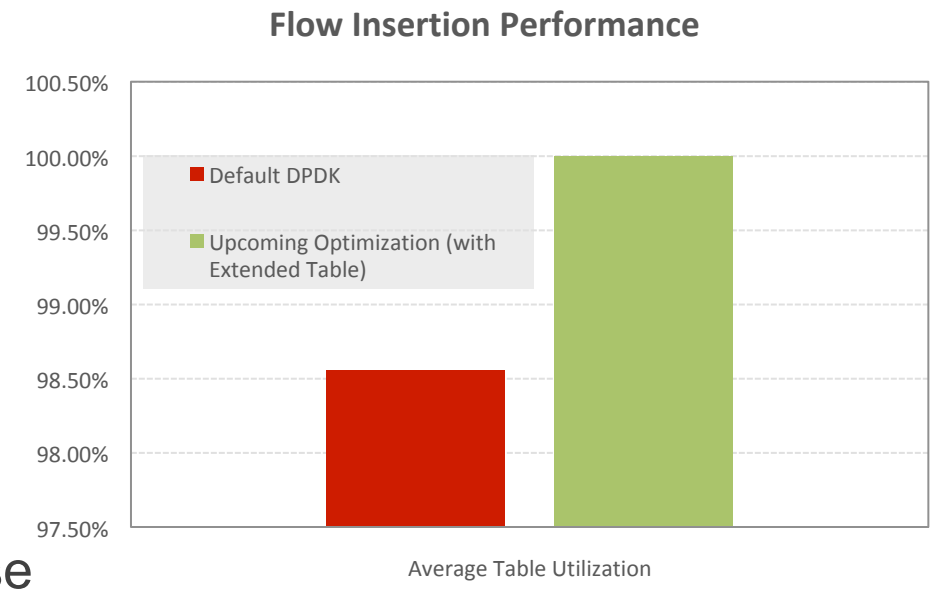


- Hardware monitors cache lines.
- When data conflict is detected, execution is rolled back

RTE Hash Upcoming Optimizations for V18.11

Add Extended Table Concept to Handle Insertion Failures





- Native Cuckoo Hash has a small percentage of insertion failure.
- Added a hierarchical approach for an extended table in case of insertion failure:
  - Dropping flows is not acceptable for telco workloads
  - Run-time flow table resizing is too slow for workload.

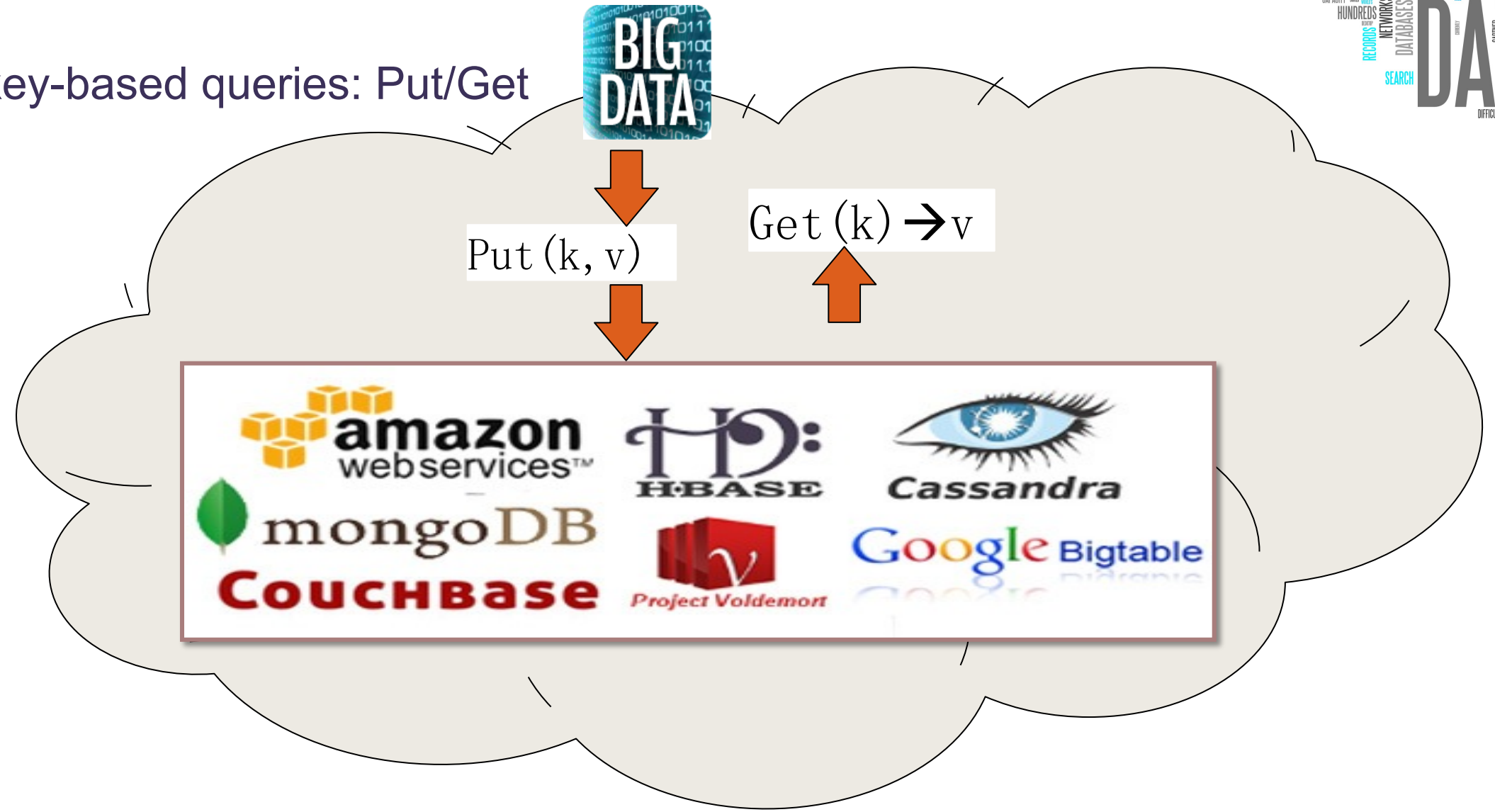
Provide 100% Insertion Guarantee needed to Support Telco Workloads

Agenda

- ✓ Overview of Flow Classification Libraries in DPDK
  - ✓ Hash Library
  - ✓ EFD (Elastic Flow Distributor) Library
  - ✓ Membership Library
- ✓ Recent optimizations for rte_hash in DPDK v18.08
- ✓ Upcoming optimizations for rte_hash targeting DPDK v18.11
- Cloud Workloads and Membership Library Research Direction

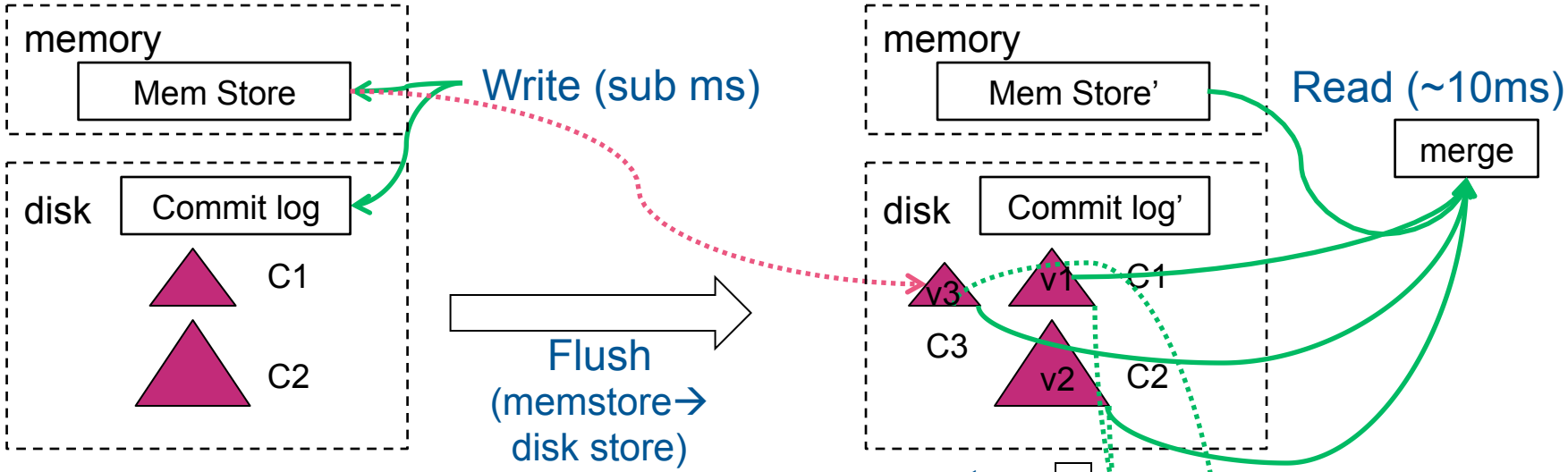
NoSQL Big Data stores

- key-based queries: Put/Get

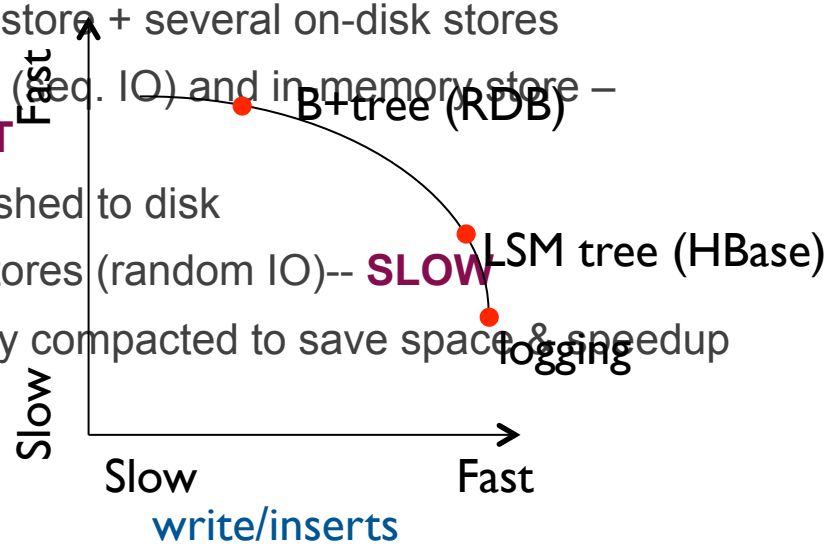


Log Structured Merge (LSM) Trees – High Level

[O'Neil, Acta Informatica'96]

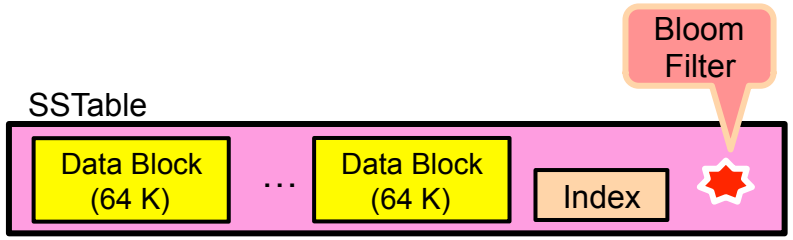


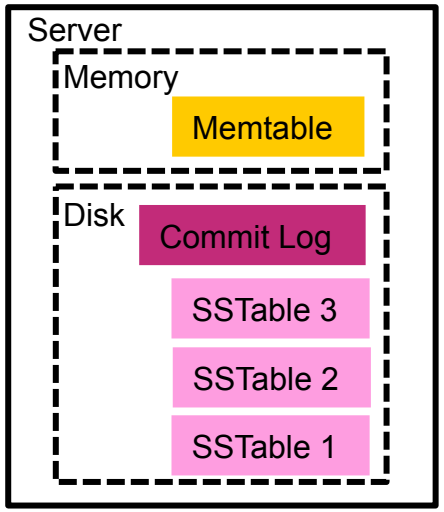
- LSM Tree = a in-memory store + several on-disk stores
- Writes go to a commit log (seq. IO) and in-memory store – update not in-place, **FAST**
- Memstore periodically pushed to disk
- Reads go to mem+disk stores (random IO)-- **SLOW**
- On-disk stores periodically compacted to save space & speedup read



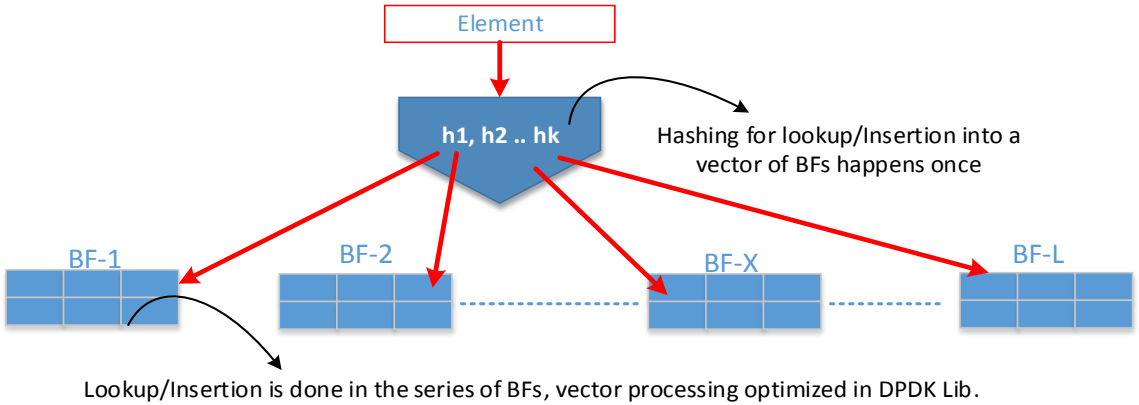
Using Membership Library (Vector Bloom Filter)

- Problem: Reads can cause a lot of disk access for all the SSTables not in memory
- Solution: Bloom filter, a space efficient probabilistic data structure to test element membership in a set
- Most lookups for non-existent rows or columns do not need to touch disk



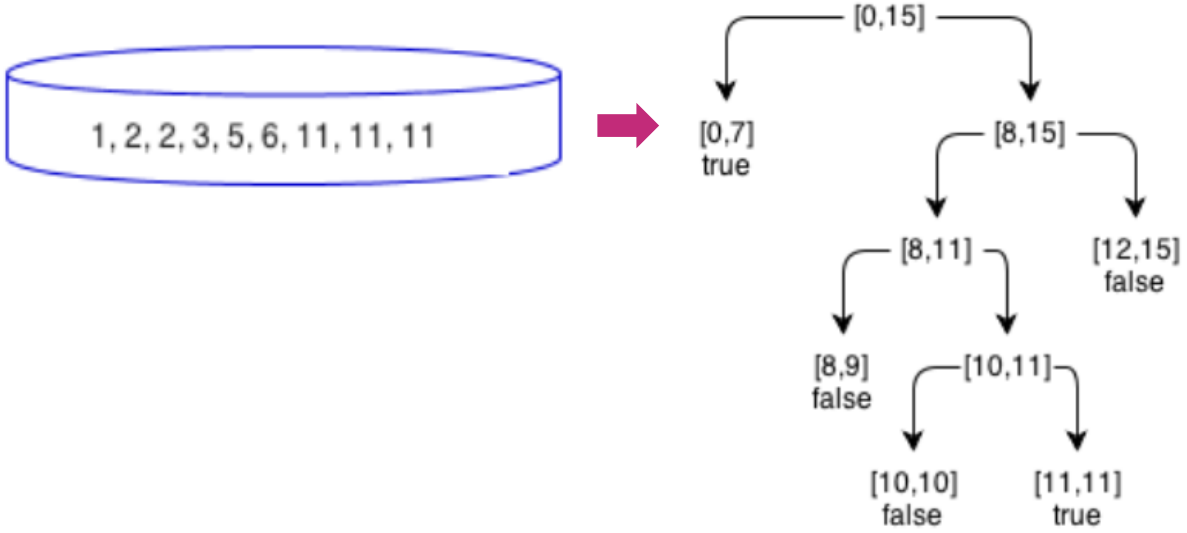


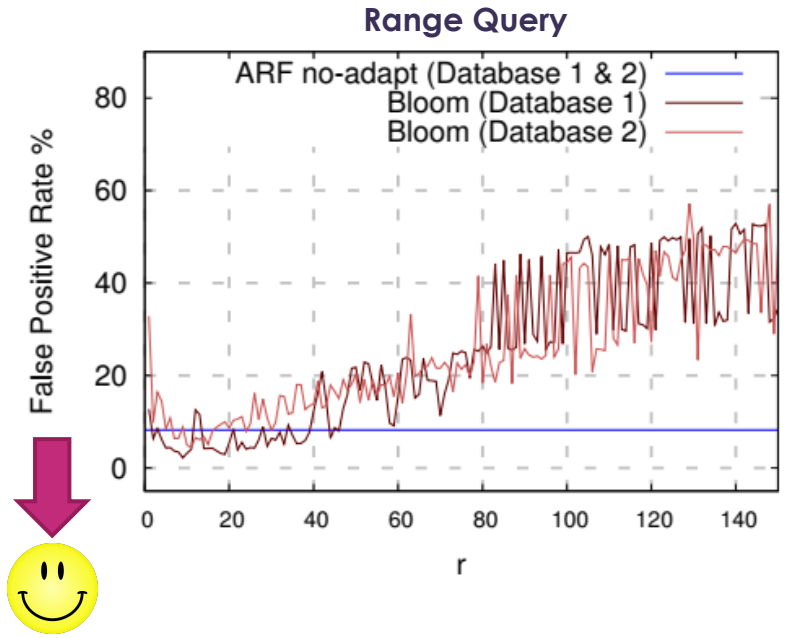
DPDK Membership Library provides an optimized library for vector parallel processing of many bloom filters simultaneously



Extend Membership Library to Range Filters

- Problem: Range queries is a very important workload for bigdata K-V stores. No current technology is optimized to efficiently handle range queries.
- Can we extend membership library to support range filters ??





Summary – Call for Action

- An overview of flow classification libraries in DPDK, and when to use each library.
  - Please try testing Membership and EFD libraries in your workload.
  - For certain networking applications and workloads they will outperform hash tables.

- Recent optimizations RTE-Hash library to support R/W concurrency released in DPDK V18.08, working towards proposing extended table design in V18.11
  - Please review the patch with the design and provide feedback.
  - Please point out gaps where the RTE-hash doesn't fit your workload for flow tables.

- Research direction of cloud workload and Big Data K-V store
  - Looking for collaborators and and developers interested in optimizing their Cloud Workloads using DPDK libraries.

Questions?

SAMEH.GOBRIEL@INTEL.COM