



Flow Classification Optimizations in DPDK

Sameh Gobriel & Charlie Tai - Intel

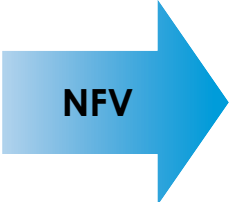
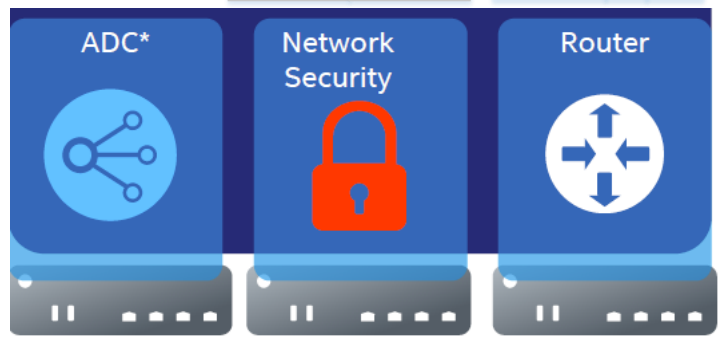
DPDK US Summit - San Jose - 2016



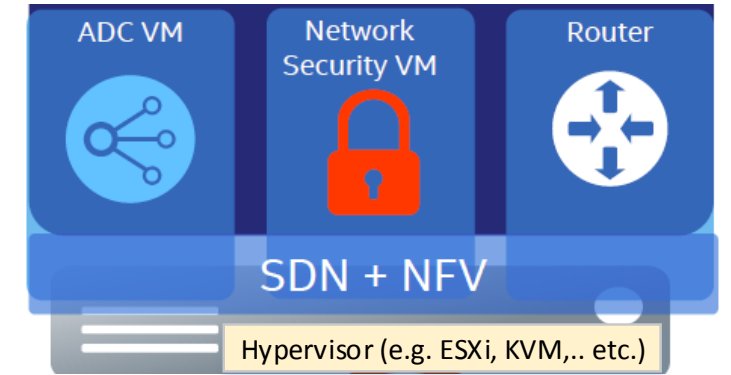
- ▶ Flow Classification in DPDK
- ▶ Cuckoo Hashing for Optimized Flow Table Design
- ▶ Using Intel Transactional Synchronization Extensions (TSX) for scaling Insert performance
- ▶ Using Intel AVX instructions for scaling lookup performance
- ▶ Research Proof of Concept: 2 level lookup for OVS Megaflow Cache

Flow Classification on Network Appliances vs General Purpose Server H/W

Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out



Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out
Flow	Target	Flow	Action	Flow	In	Out



Flow Classification Implemented on General Purpose Processors

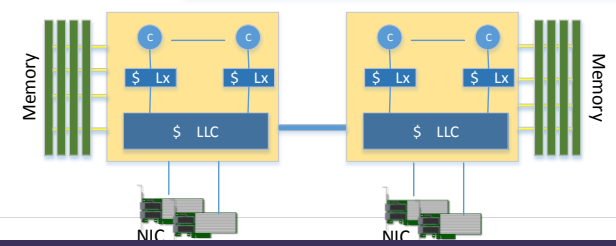


TEM/OEM



ASIC, DSP,

Monolithic Purpose-built Boxes

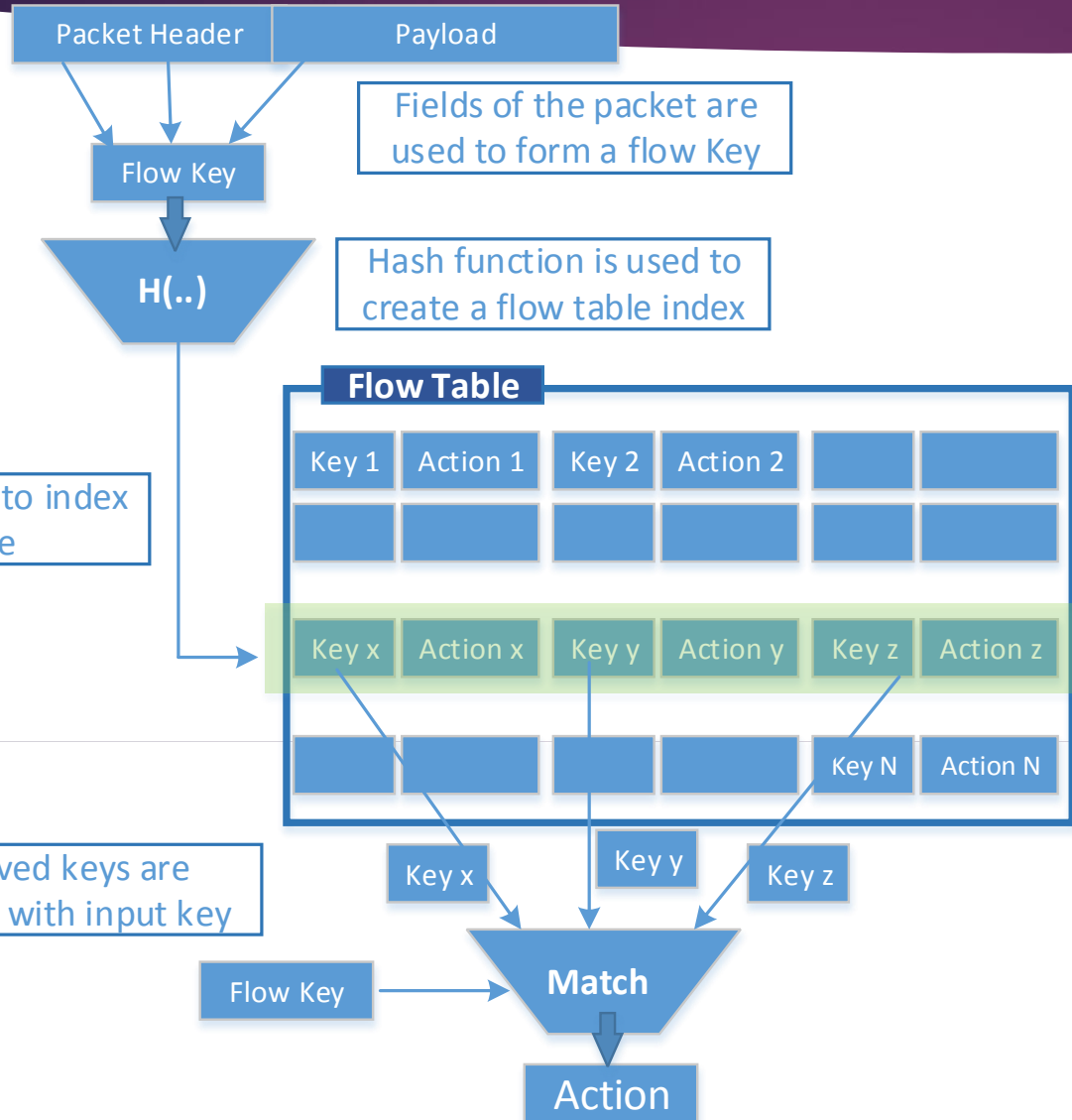


Networking VMs on Standard Servers

- Network appliances use purpose-built H/W & ASICs (e.g., TCAM) for flow classification
- Cost & power consumption are limiting factors to support large number of flows

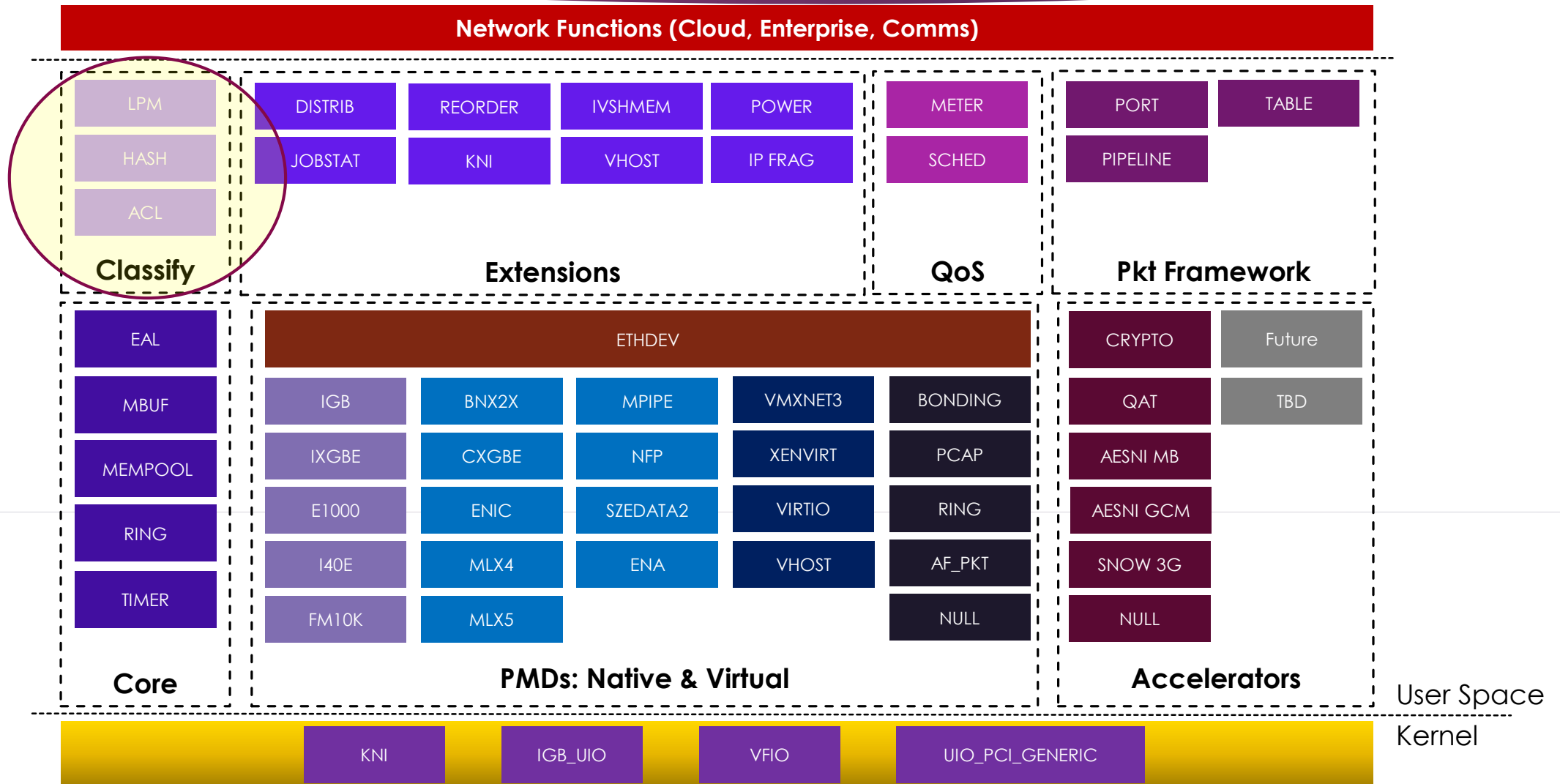
- General purpose processors with Cache/memory hierarchy can support much larger flow tables.
- Multicores architecture provide a scalable competitive flow classification performance.

Metrics for Good Flow Table Design

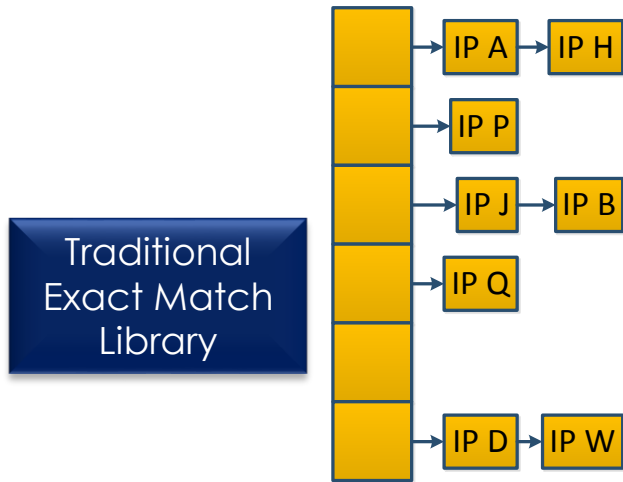


- 1. Higher Lookup Rate** = Better throughput & latency
- 2. Higher Insert Rate** = Better Flow update & Table Initialization
- 3. Efficient Table Utilization** = More Flows

DPDK Framework

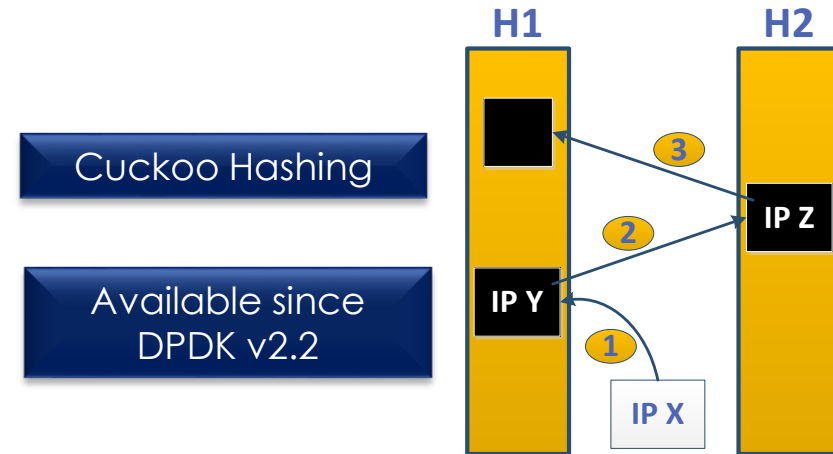


RTE-Hash Exact Match Library



Traditional Exact Match Table library:

- relies on a “sparse” hash table implementation
- Simple exact match implementation
- Significant performance degradation with increased table sizes.



Cuckoo Hashing - Better Scalability:

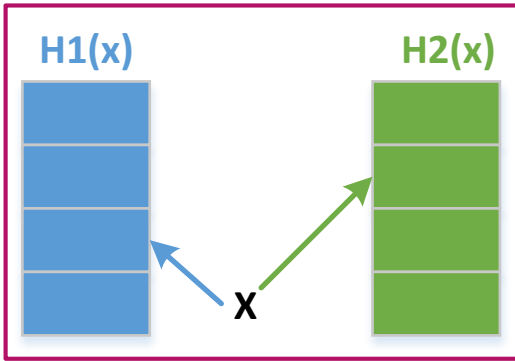
- Denser tables fit in cache.
- Can scale to millions of entries.



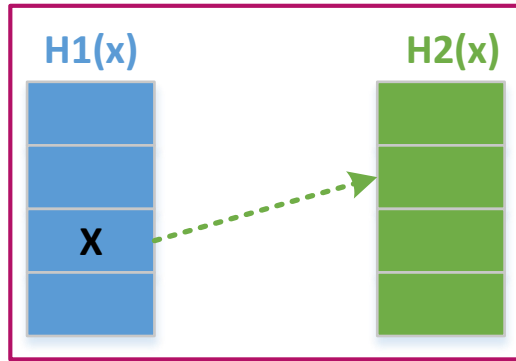
Cuckoo Hashing High Level Overview



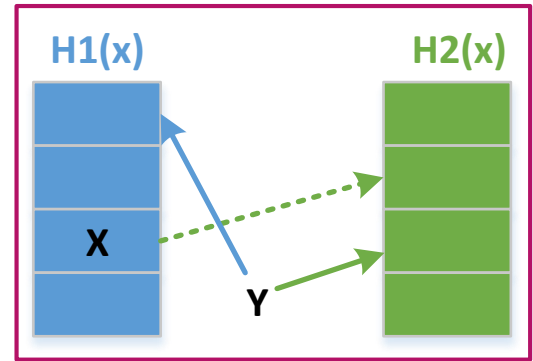
1



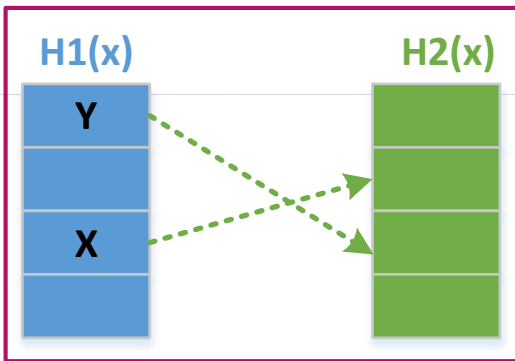
2



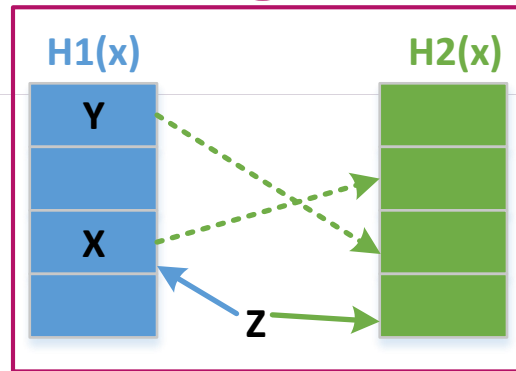
3



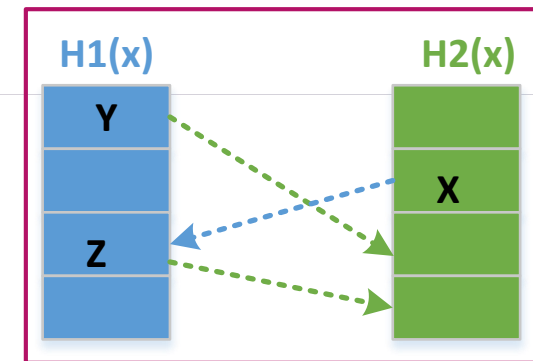
4



5



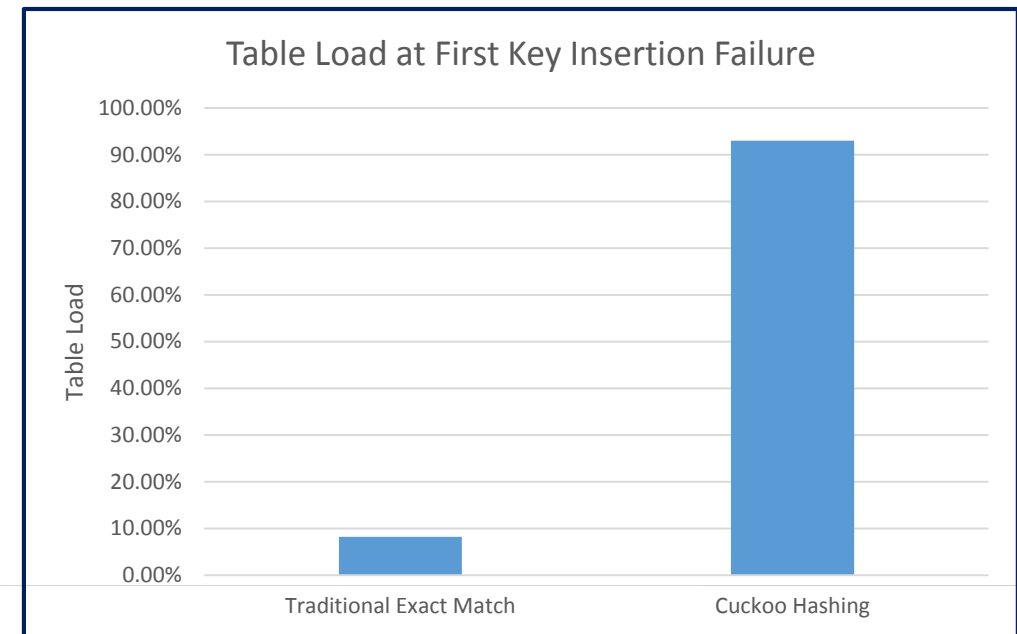
6



Cuckoo Hashing Performance Benefits



- ▶ Cuckoo Hashing allows for more flows to be inserted in the flow table
- ▶ RTE-hash can be used to support flow table with millions of keys (e.g. 64M – 5 tuple keys) that fits in the CPU cache.



Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
Hyper-Threading: disabled

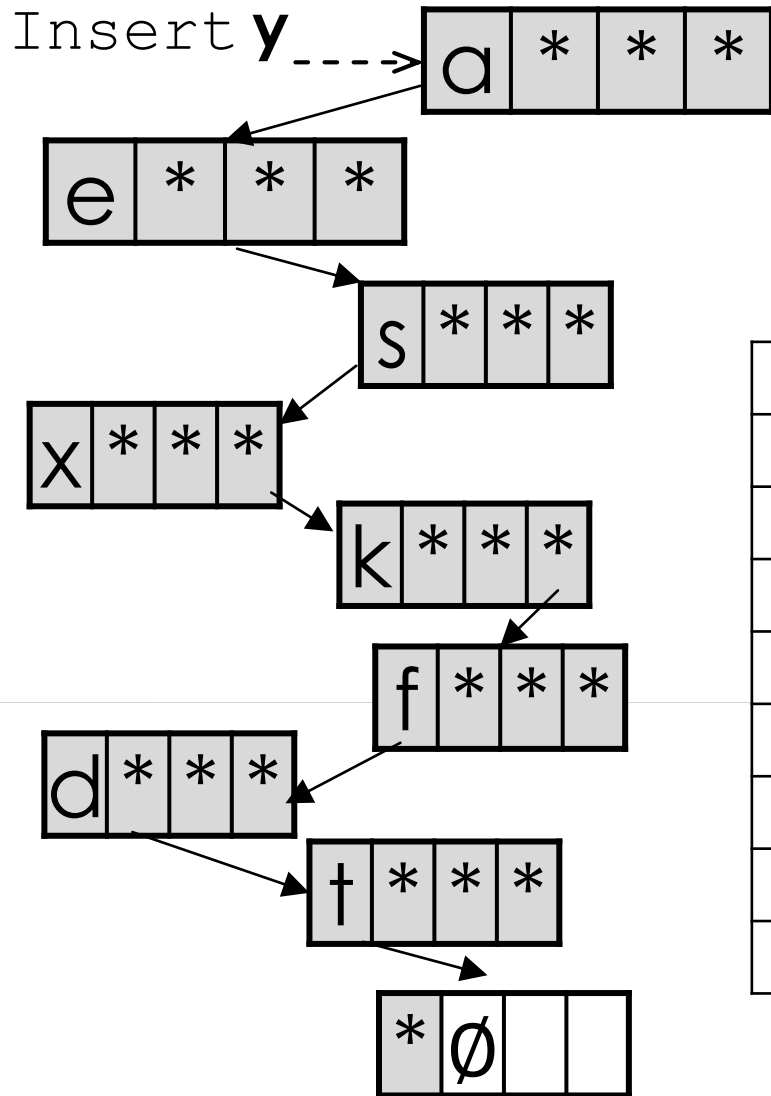
Code Snippet for RTE-hash API



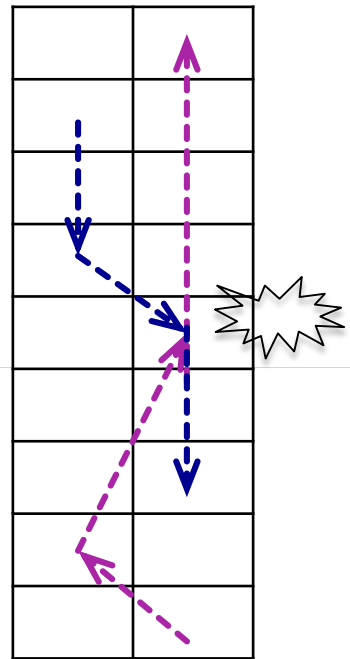
- ▶ `struct rte_hash *rte_hash_create (const struct rte_hash_parameters *params)`
- ▶ `int rte_hash_add_key_data (const struct rte_hash *h, const void *key, void *data)`
- ▶ `int rte_hash_lookup_data (const struct rte_hash *h, const void *key, void **data)`
- ▶ `int rte_hash_lookup_bulk_data (const struct rte_hash *h, const void **keys, uint32_t num_keys, uint64_t *hit_mask, void *data[])`

Reference: http://dpdk.org/doc/api/rte__hash_8h.html

Long Cuckoo Paths & Multiple Concurrent Writers



cuckoo path:
 $a \rightarrow e \rightarrow s \rightarrow x \rightarrow k \rightarrow f \rightarrow d \rightarrow t \rightarrow \emptyset$ (**9 writes**)



One Insert may move a lot of items especially at high table occupancy

← collision

Collision happens when multiple writers have intersecting Cuckoo Paths

Flow-Table Insert Performance Optimizations

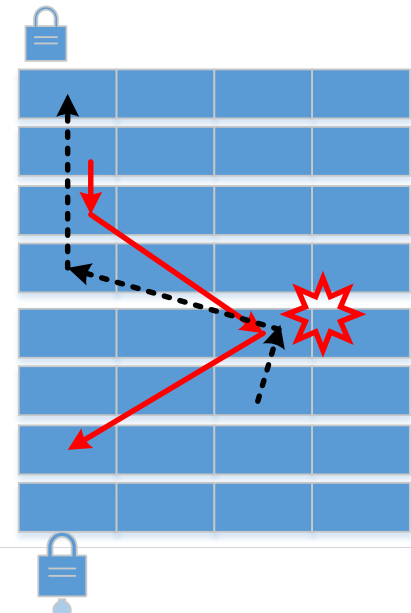


Insert Performance Optimizations

Make Use of IA Hardware Features

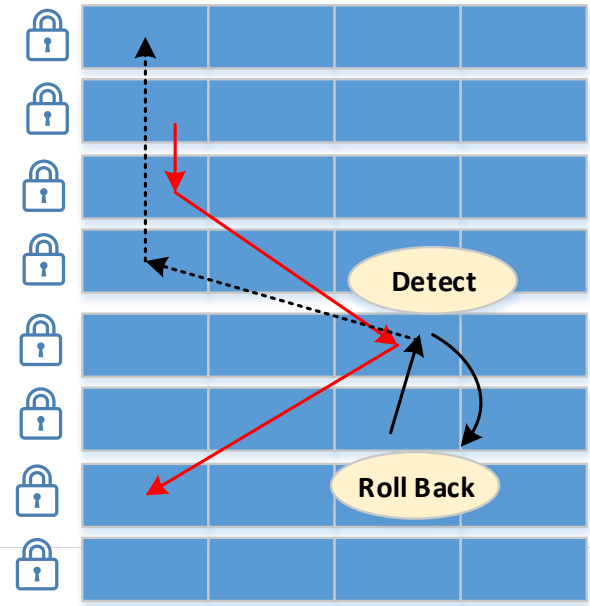
Minimize Critical Section

Traditional 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

Flow-Table Insert Performance Optimizations



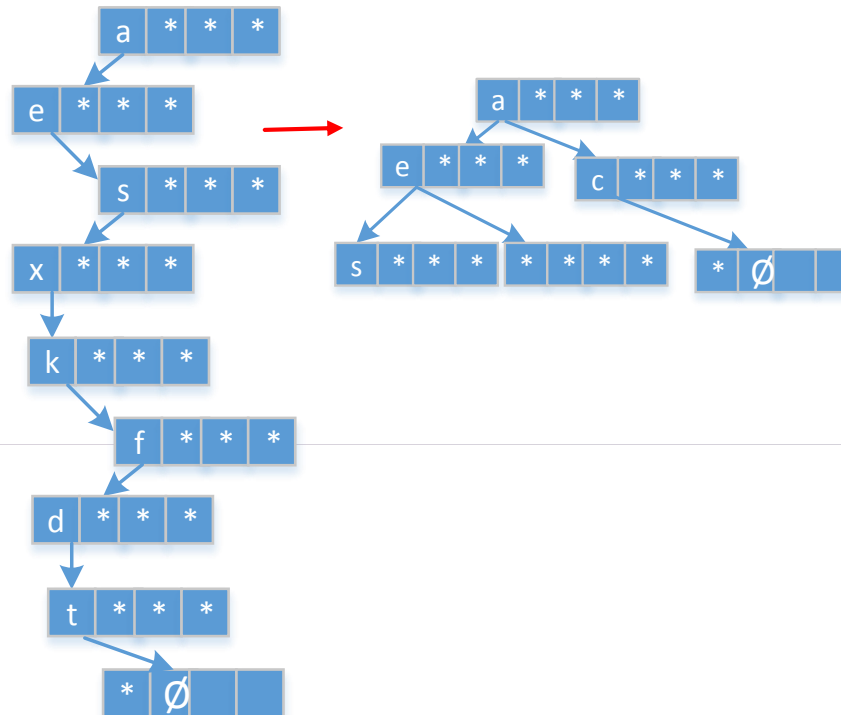
Insert Performance Optimizations

Make Use of IA Hardware Features

Minimize Critical Section

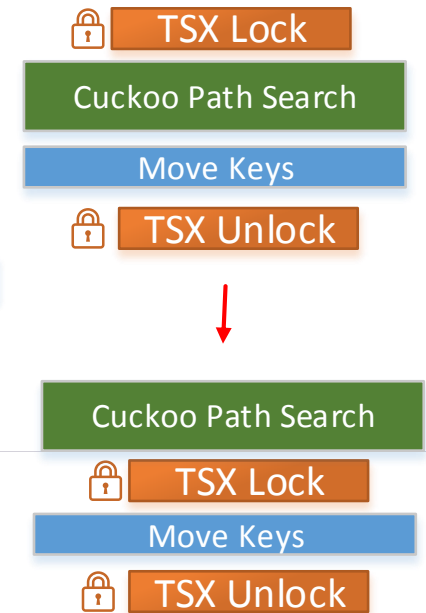
1

Depth First Search → Breadth First Search

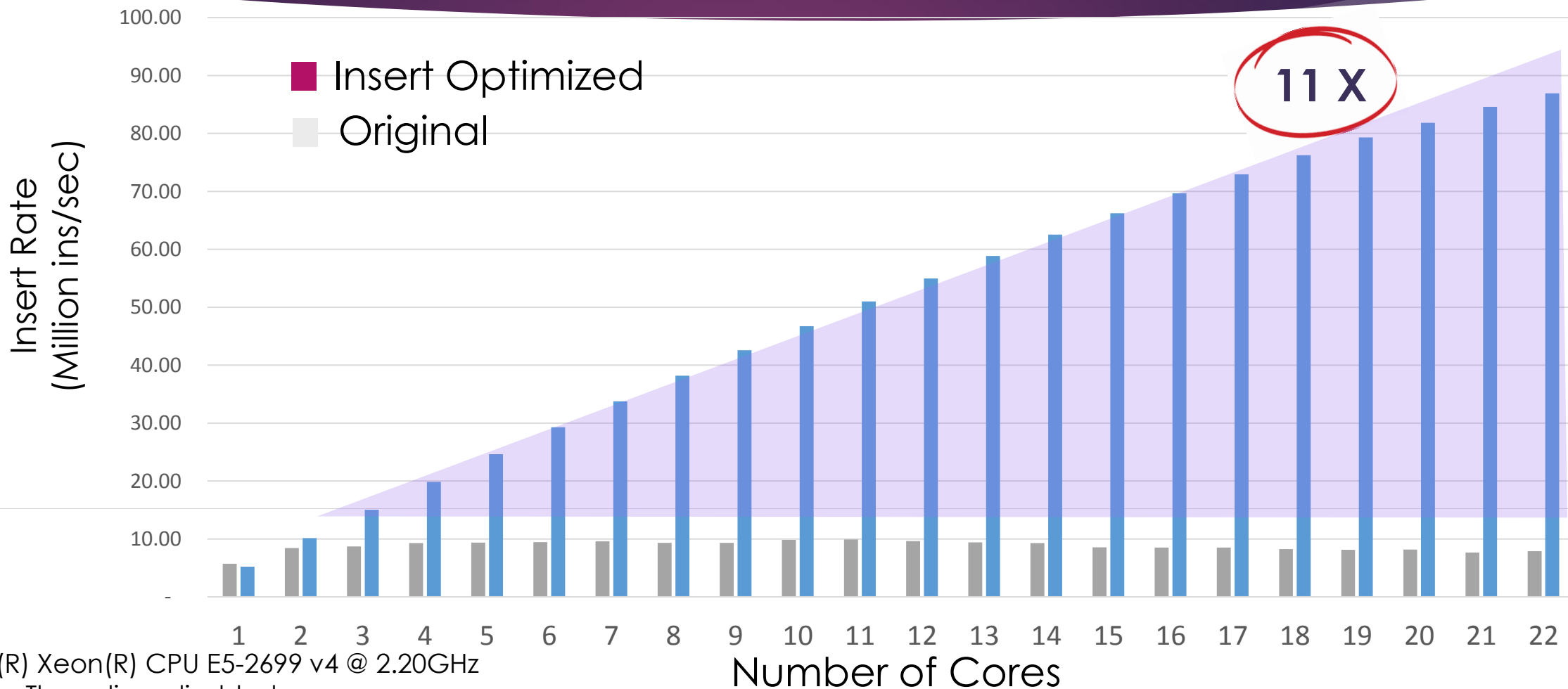


2

Split Path Search from Keys Movement



Summary of Insert Optimizations Results



Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
Hyper-Threading: disabled

Insert Performance Linearly Scalable with Number of Cores

Code Snippet for RTE-hash with TSX (DPDK V16.07)



```
#define RTE_HASH_EXTRA_FLAGS_MULTI_WRITER_ADD    0x02
/* Default behavior of insertion, single writer/multi writer */
struct rte_hash_parameters {
    ...
    uint8_t extra_flag;
};
rte_hash_parameters.extra_flag |=
    (RTE_HASH_EXTRA_FLAGS_TRANS_MEM_SUPPORT
     | RTE_HASH_EXTRA_FLAGS_MULTI_WRITER_ADD);
```

► To enjoy TSX enabled multiwriter.

Reference: http://dpdk.org/doc/api/rte__hash_8h.html

Flow-Table Lookup Performance Optimizations



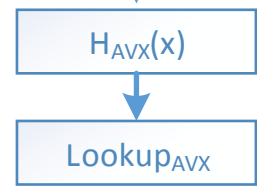
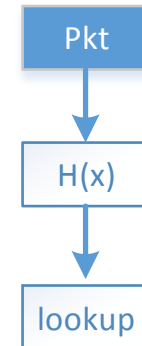
Lookup Performance Optimizations

Make Use of IA Hardware Features

Minimize Implementation Overhead

1

Use AVX Instructions

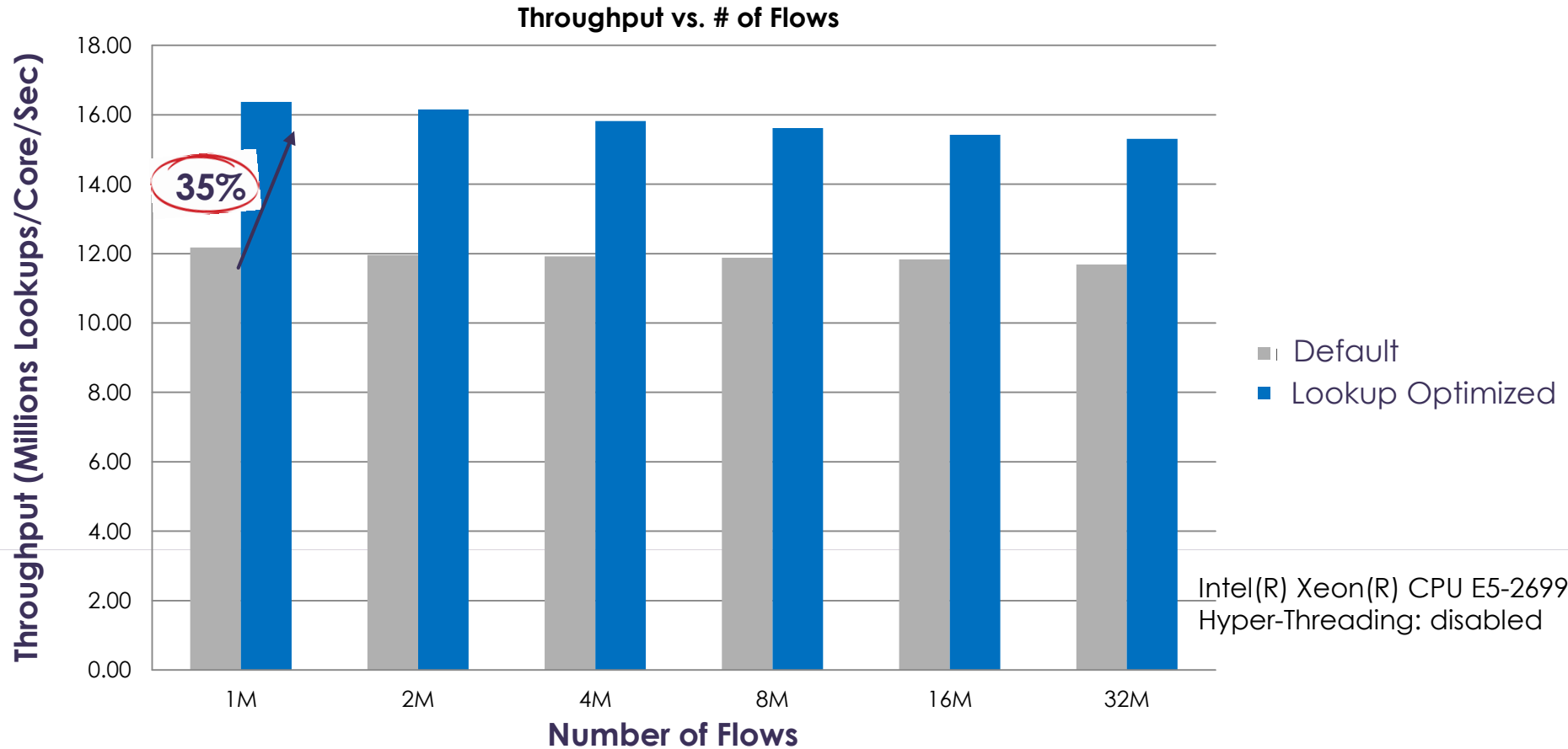


2

Minimize Overhead

1. Prefetching of Keys in Cache.
2. Inline Functions
3. Lookup Pipelining

Summary of Lookup Optimizations Results

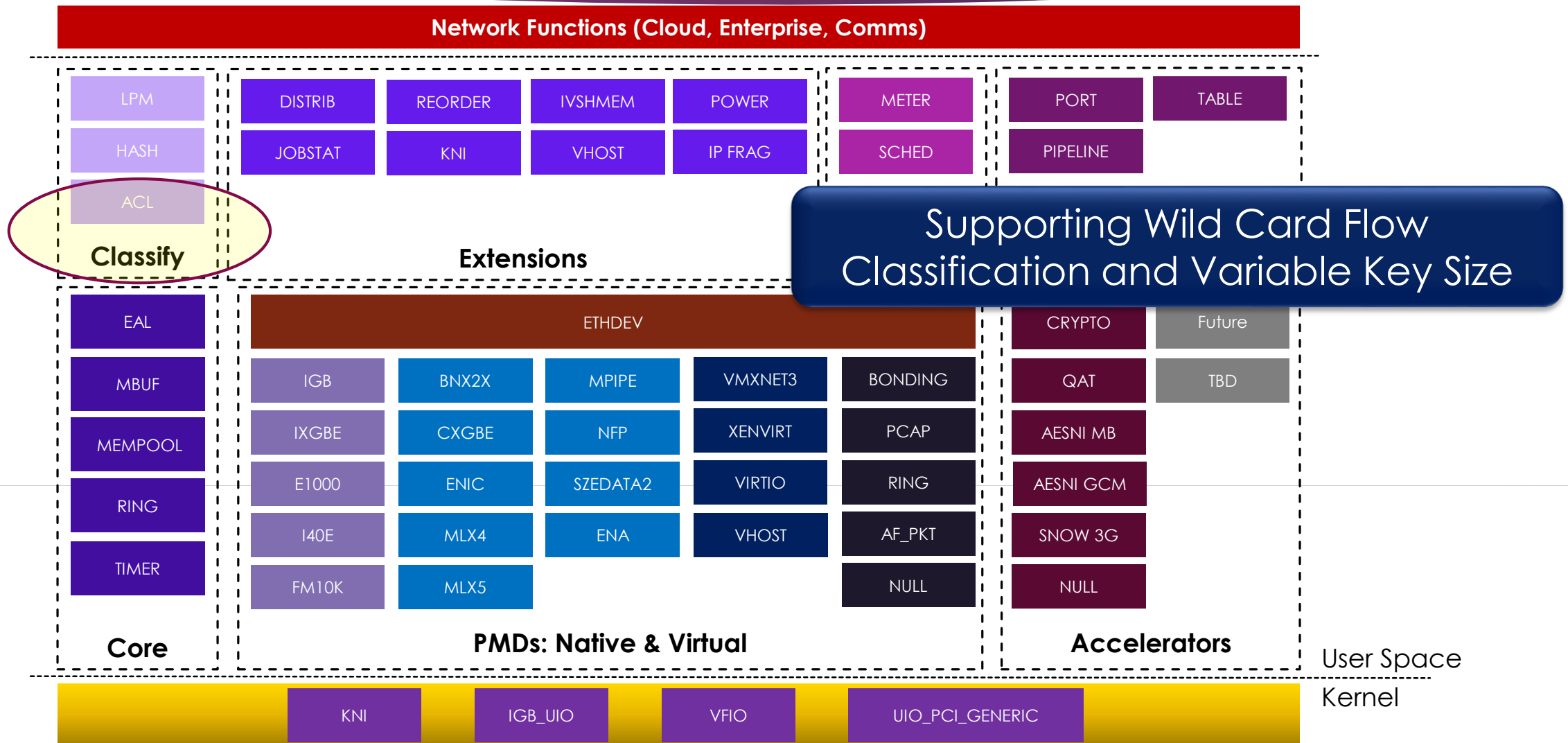


~35% Improved Lookup Throughput

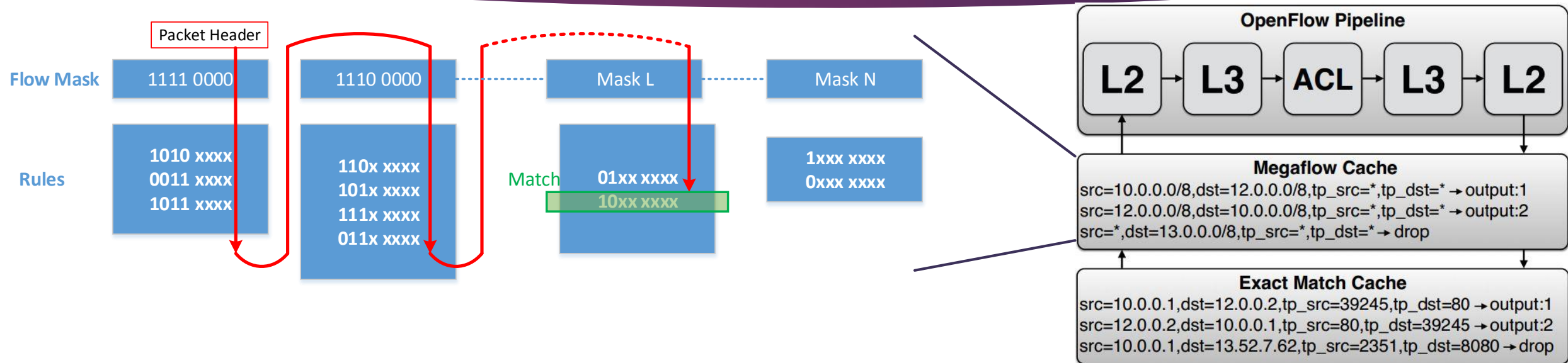
Code Snippet for RTE-hash with AVX (Targeting DPDK V16.11)



DPDK Framework



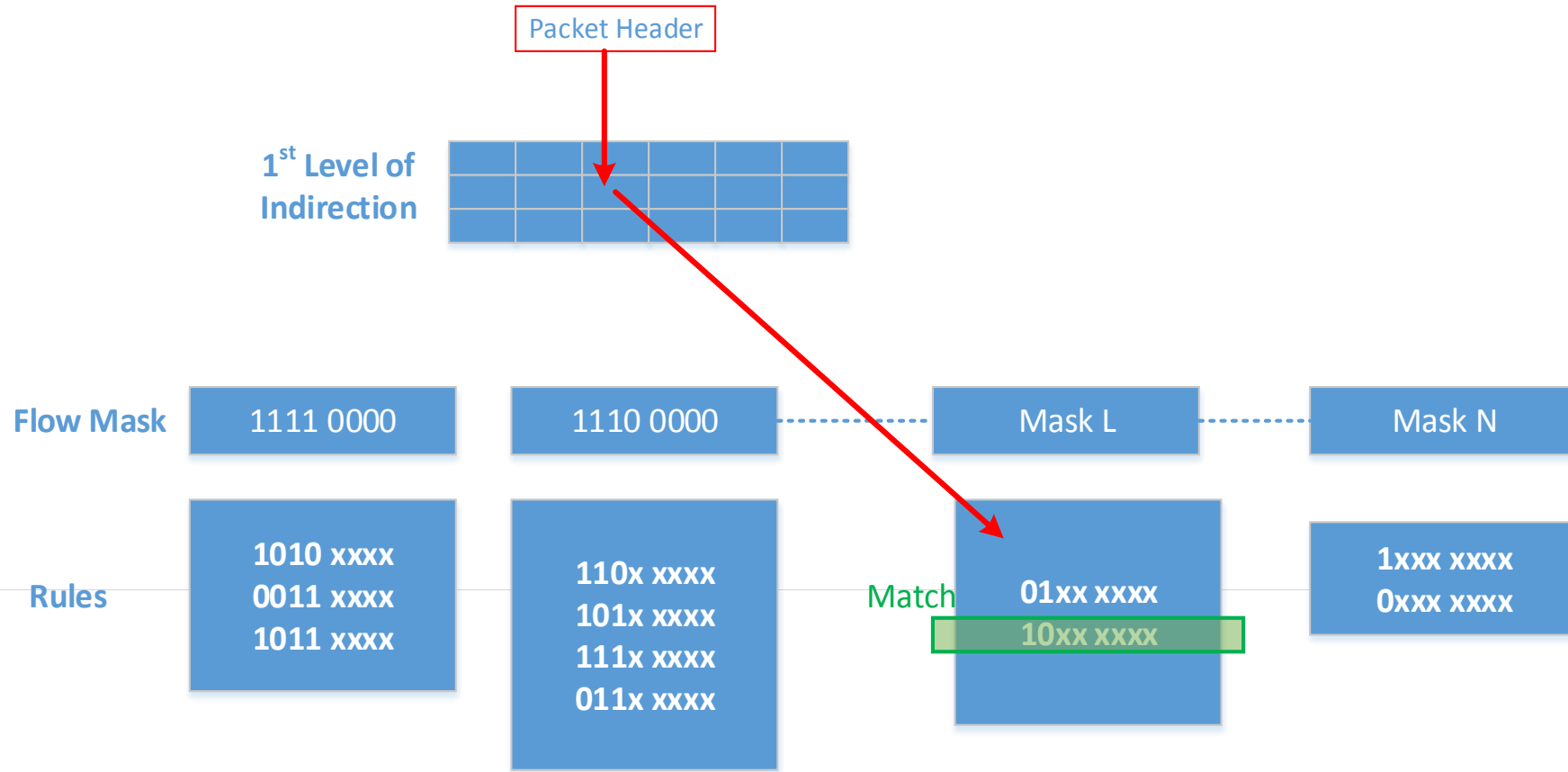
POC: Open vSwitch Flow Lookup



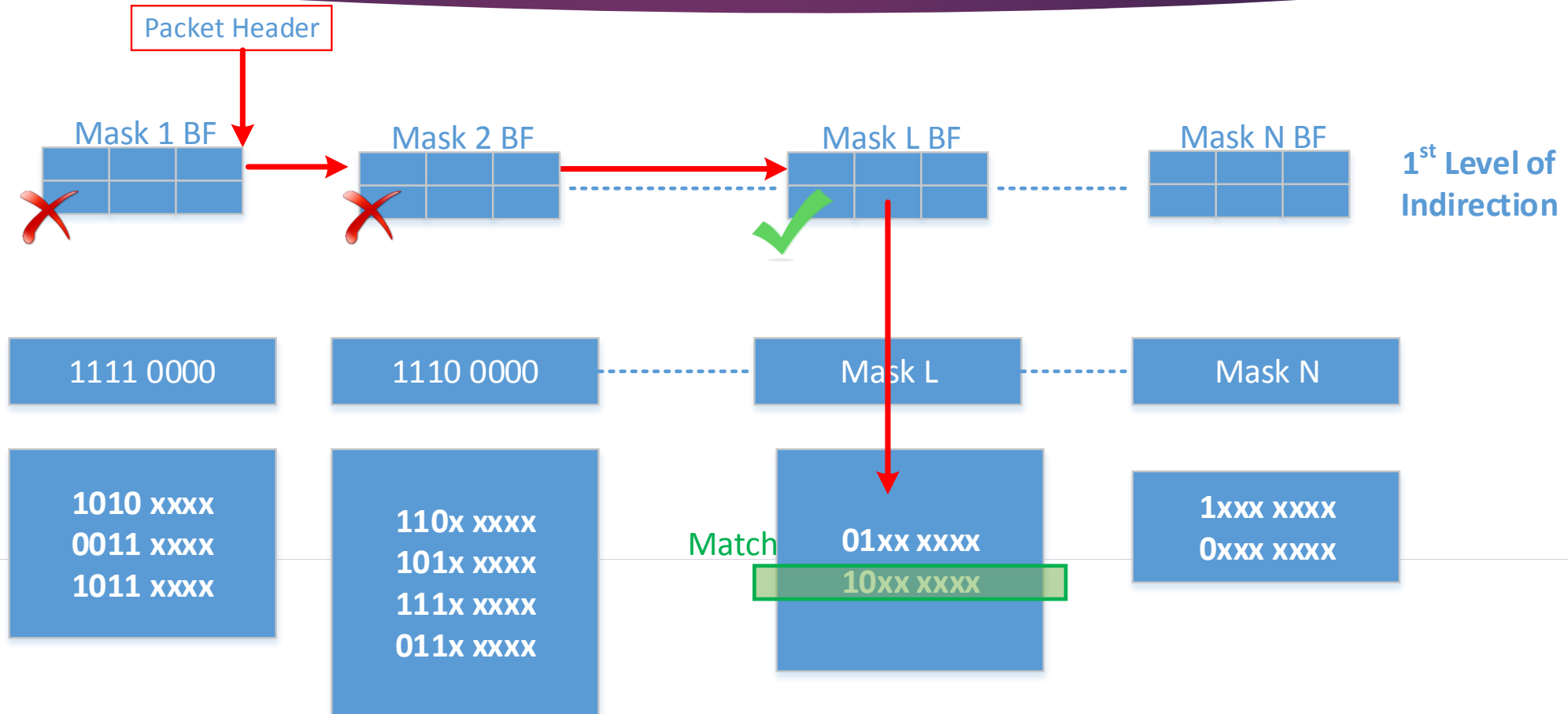
1. Set of disjoint sub-table
2. Rule is only inserted into one sub-table (lookup terminates after first match)
3. Lookup is done by sequentially search each sub-table until a match is found

Instead of L sequential lookups → What if we know which sub-table to hit

OVS with Two Layer Lookup

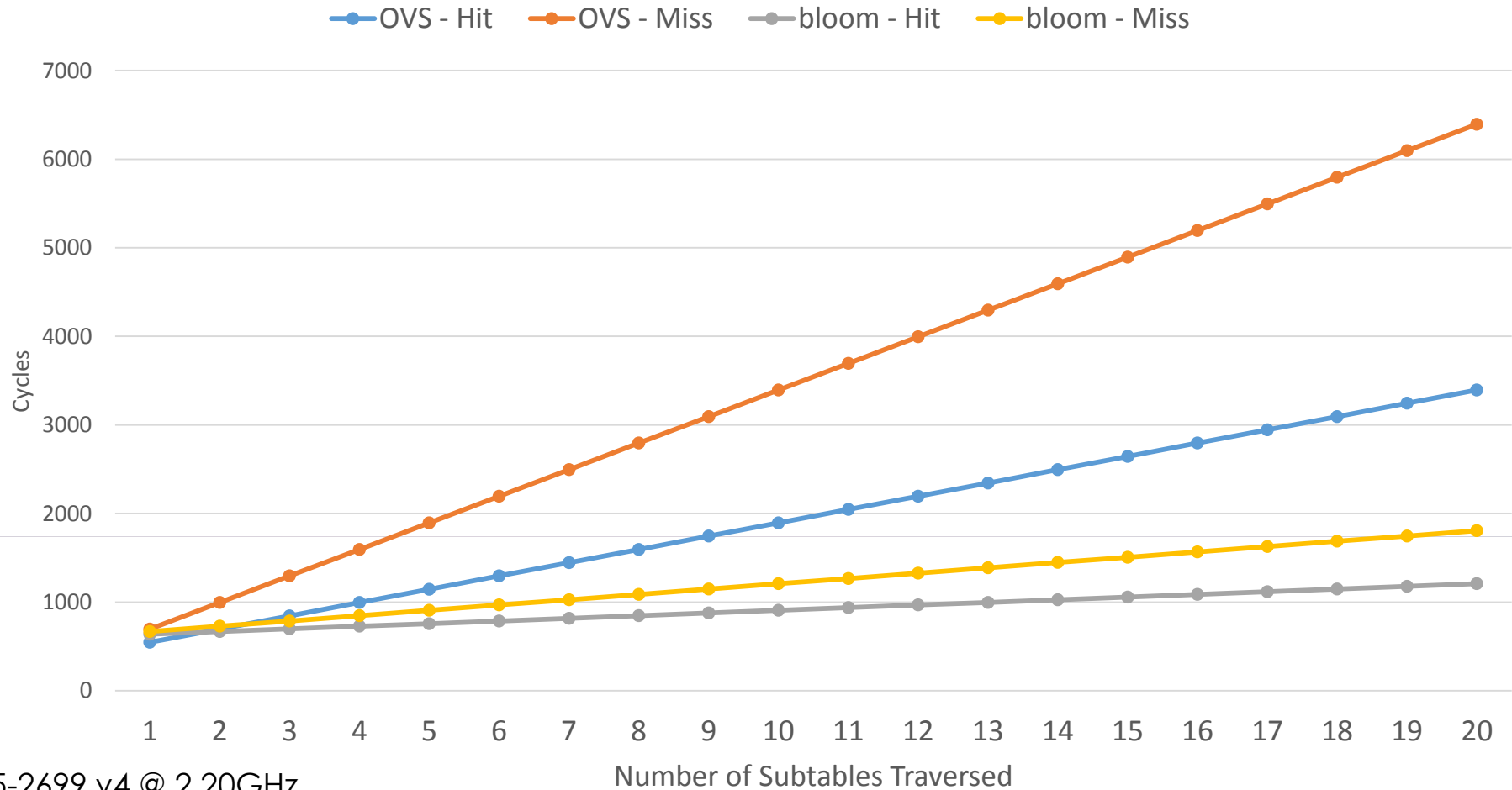


Bloom Filter as 1st Level of Indirection



L Lookups → L Bloom Filters + 1 lookup

2 Level Lookup Preliminary Performance Results



Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
Hyper-Threading: disabled

Legal Disclaimers



No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or learn more at intel.com.

© 2016 Intel Corporation. Intel, the Intel logo, Intel. Experience What's Inside, and the Intel. Experience What's Inside logo are trademarks of Intel. Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Questions?

Sameh Gobriel

sameh.gobriel@intel.com