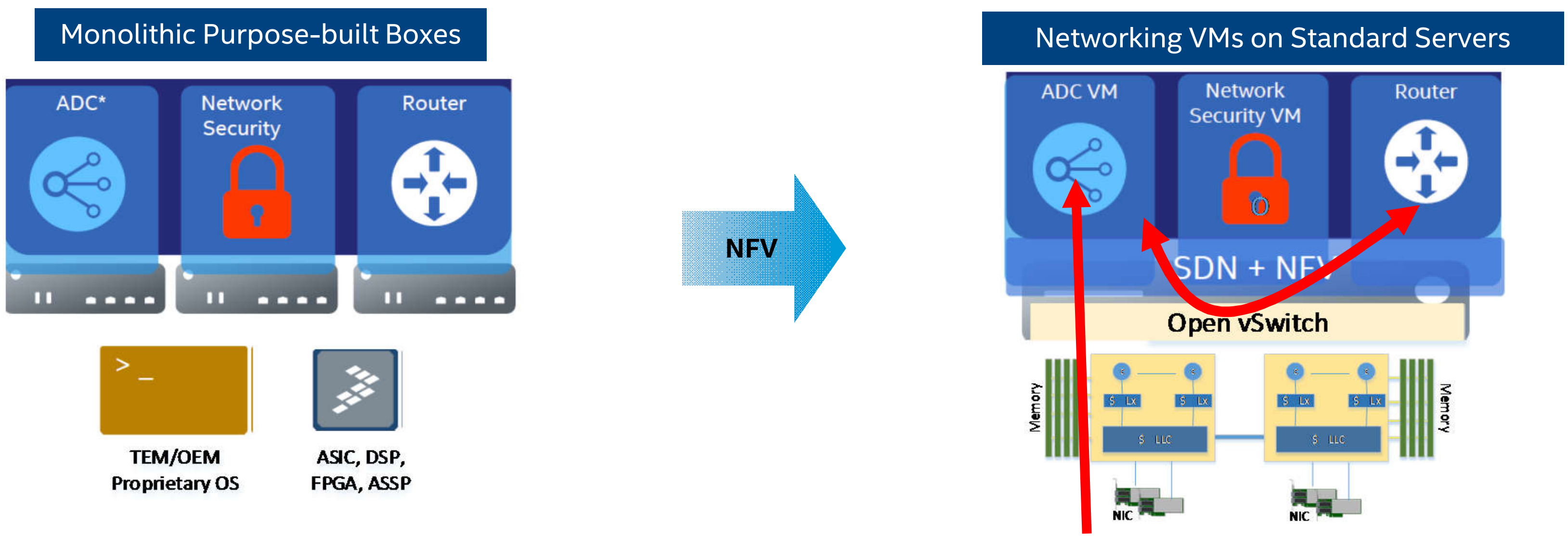
OvS Lookup Optimization

Using Two-Layer Table Lookup

Sameh Gobriel and Charlie Tai

Intel Labs

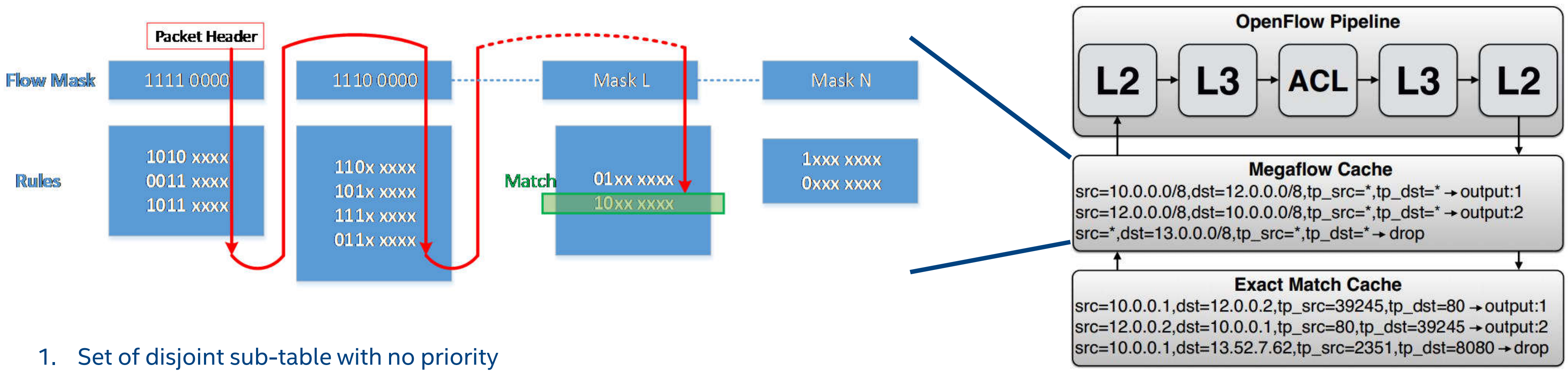
OvS De Facto Virtual Switch for NFV Environments



- 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.

Open vSwitch Flow Lookup



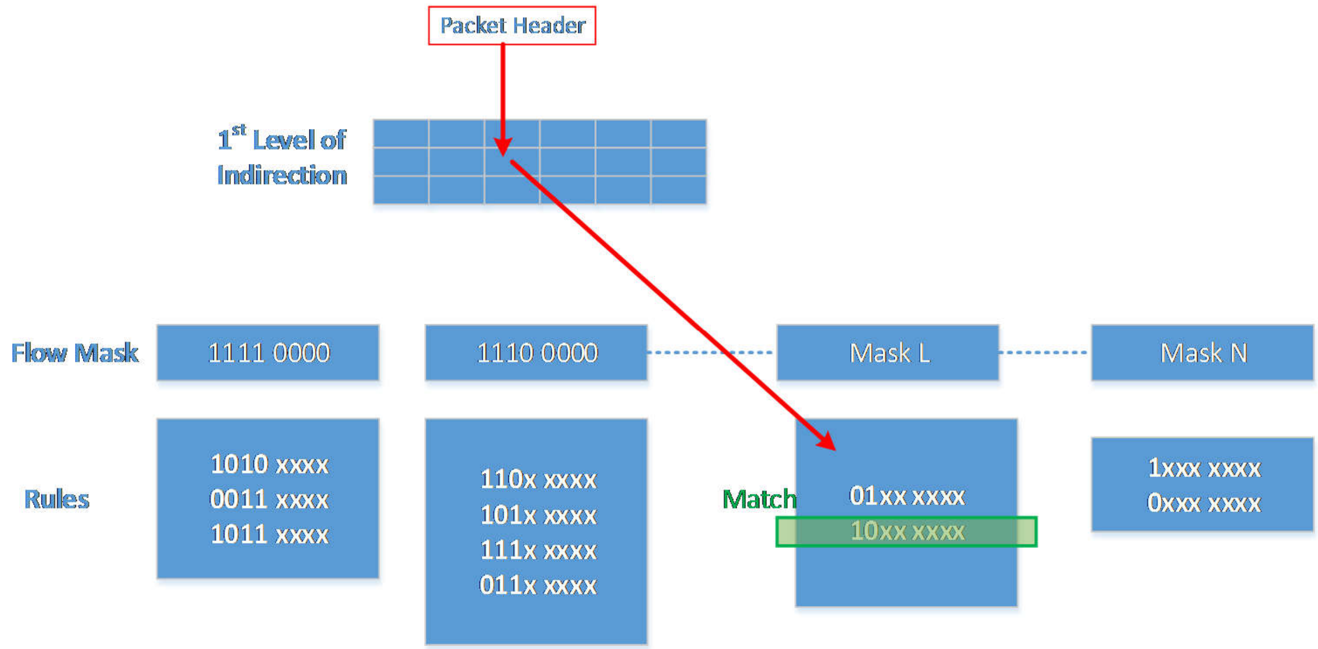
1. Set of disjoint sub-table with no priority
2. Rule is only inserted into one sub-table (lookup terminates after first match)
3. Lookup is done by sequentially search each sub-table until a match is found

OvS Flow Classification is a bottleneck

fast_path_processing	54.3%
dpcls_lookup	53.6%
netdev_flow_key_hash_in_mask	39.3%
dpcls_rule_matches_key	7.1%
zero_rightmost_1bit	0.0%
pvector_cursor_next	0.0%

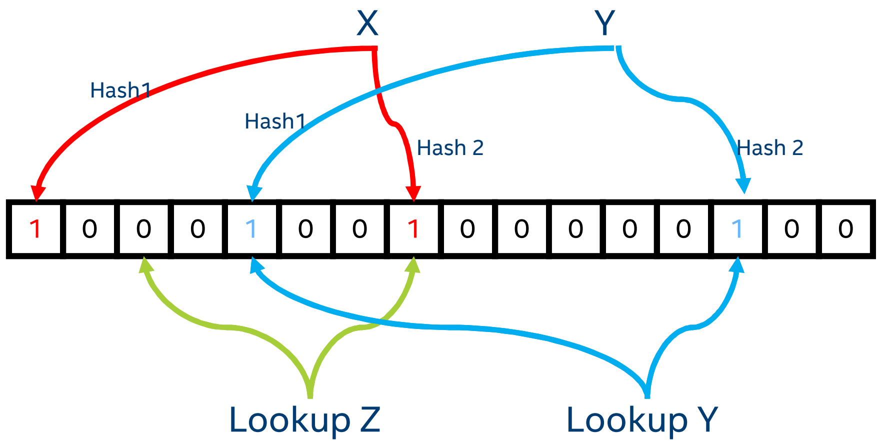
Fig. Vtunes OVS flow lookup process (bypass EMC). Test case: 20 sub-tables, each has 100 rules.

Two Layer Table Lookup Abstraction for MFC



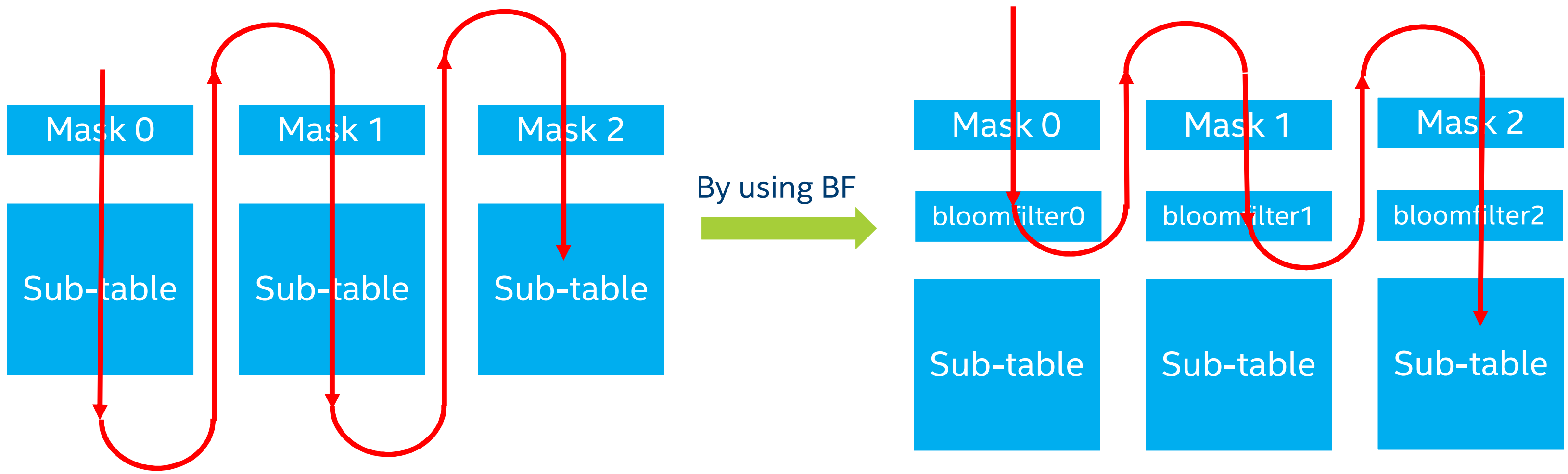
L Lookups → 1 lookup + 1st Level Indirection Overhead

Bloom Filter – Background



- With certain false positive rate, bloom filter is used to check if a variable (x,y,z) is a member. Member means the variable has been inserted already.
- We can use bloom filter to check if a flow is inside a sub-table or not, before searching the sub-table.

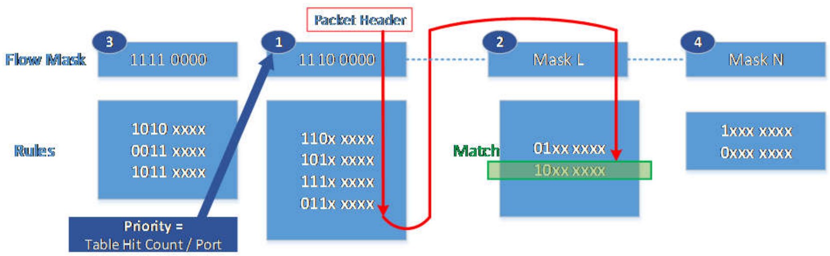
Bloom Filter – Lookup Scheme



- Before searching into sub-table, we use bloom filter to check if the masked key (sub-key) is a member of the sub-table or not.

Cycles Breakdown Assuming L-subtable traversal

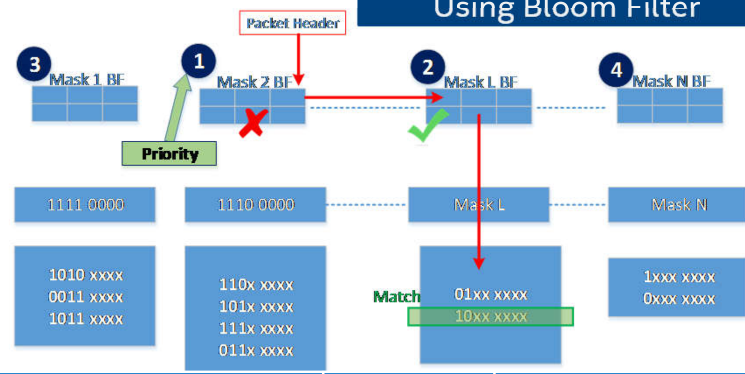
Using Current Scheme



Operation	Cycles	Repetition
I/O	210	1
Miniflow Extract	103	1
Hash For Submask	97	L
Subtable Sig. Cmp	53	L
Full Key Comparison	82	1

Hit Cycles = 395 + L * 150

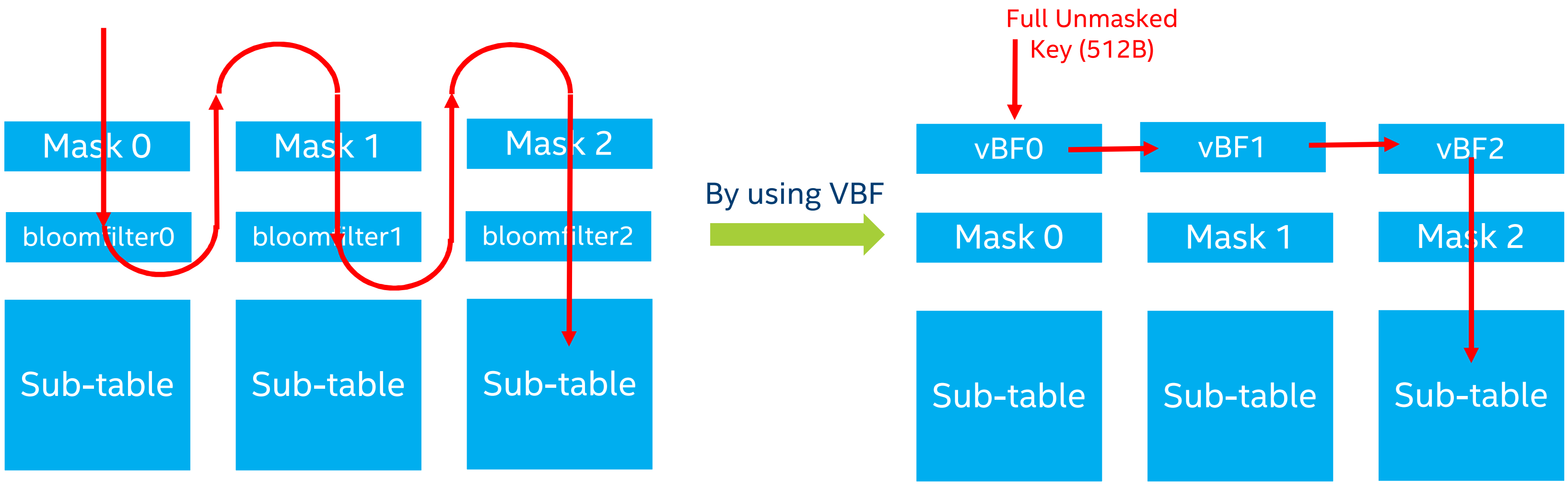
Using Bloom Filter



Operation	Cycles	Repetition
I/O	210	1
Miniflow Extract	103	1
Hash For Bloom Filter	88	L
Bloom Filter Lookup	30	L
Check Subtable Sig. Cmp	53	1
Full Key Comparison	82	1

Hit Cycles = 448 + L * 118

Vector Bloom Filter – Lookup Scheme

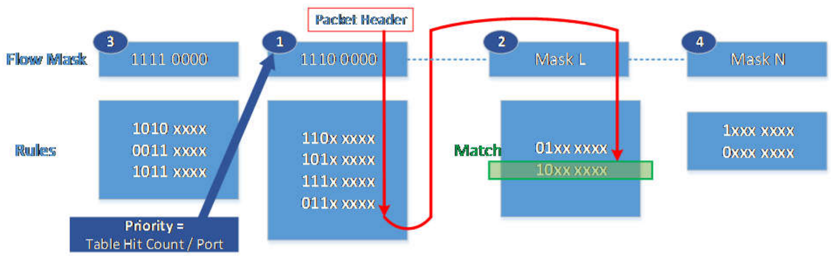


▾ fast_path_processing	54.3%
▾ dpcls_lookup	53.6%
▸ netdev_flow_key_hash_in_mask	39.3%
▸ dpcls_rule_matches_key	7.1%
▸ zero_rightmost_1bit	0.0%
▸ pvector_cursor_next	0.0%

- Vector Bloom Filter (or vBF) hashes and stores unmasked full keys (like EMC).
- vBF Filter for each sub-table store encountered full keys corresponding to rules in sub-tables
- A new flow always misses vBF (similar to EMC) but can hit a rule in the sub-table.

Vector Bloom Filter – Cost Analysis

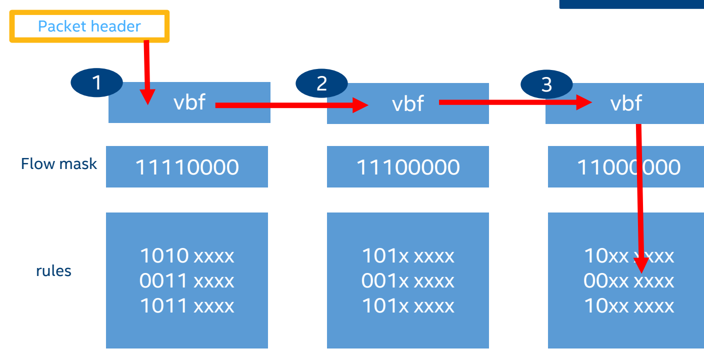
Using Original Scheme



Operation	Cycles	Repetition
I/O	210	1
Miniflow Extract	103	1
Hash For Submask	97	L
Subtable Sig. Cmp	53	L
Full Key Comparison	82	1

Hit Cycles = 395 + L * 150

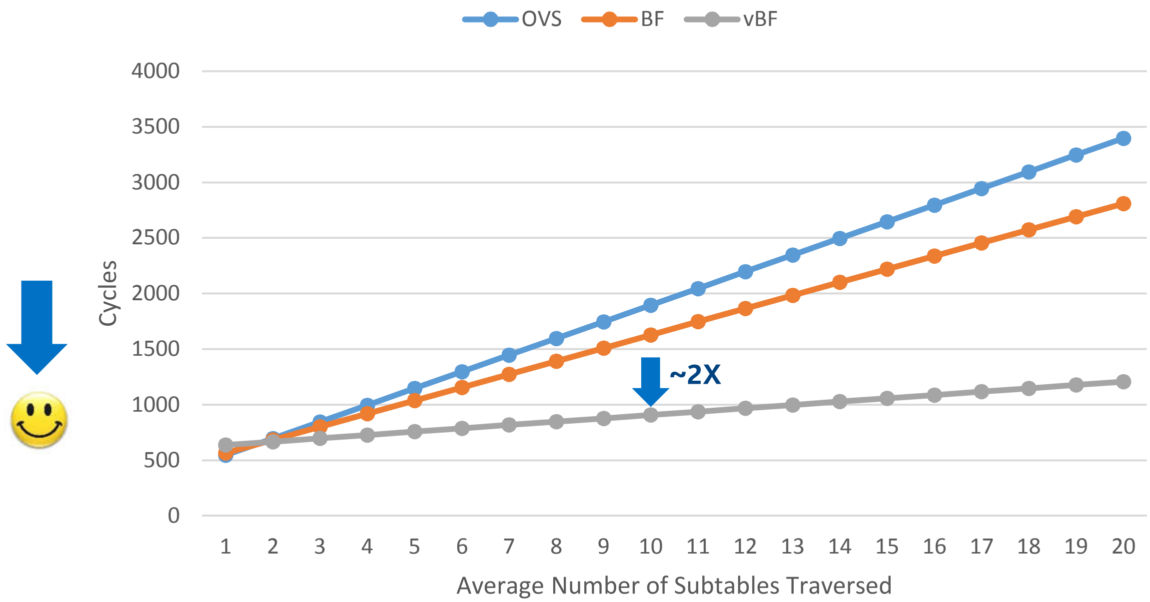
Using vBF



Operation	Cycles	Repetition
I/O	210	1
Miniflow Extract	103	1
Hash For XBloom (full Key)	159	1
Bloom Filter Lookup	30	L
Check Subtable Sig. Cmp	53	1
Full Key Comparison	82	1

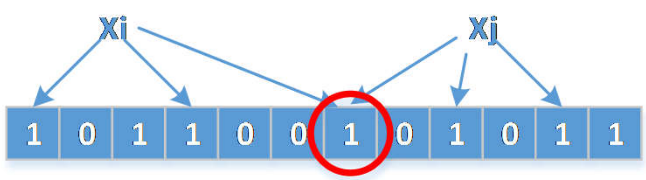
Hit Cycles = 607 + L * 30

Lookup Cycles Based on Model



Results and performance figures are for an experimental prototype and is work in progress. The result reflect specific components on a particular test, in specific systems and should not be generalized for actual products. Differences in hardware, software, or configuration will affect actual performance.
Results are generated using a model based on processing cycles of Intel Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz using OvS 2.6.0 with 20 sub-table and uniform random traffic.

Counting Bloom Filters to Handle Deletion



Start with an *m* bit array, filled with 0s.

B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in k times. If $H_i(x_j) = a$, add 1 to $B[a]$.

B	0	3	0	0	1	0	2	0	0	3	2	1	0	2	1	0

To delete x_j decrement the corresponding counters.

B	0	2	0	0	0	0	2	0	0	3	2	1	0	1	1	0

Can obtain a corresponding Bloom filter by reducing to 0/1.

B	0	1	0	0	0	0	1	0	0	1	1	1	0	1	1	0

4 bits/counter → Probability of Overflow = 6.78 E-17

Vector Bloom Filter – Results

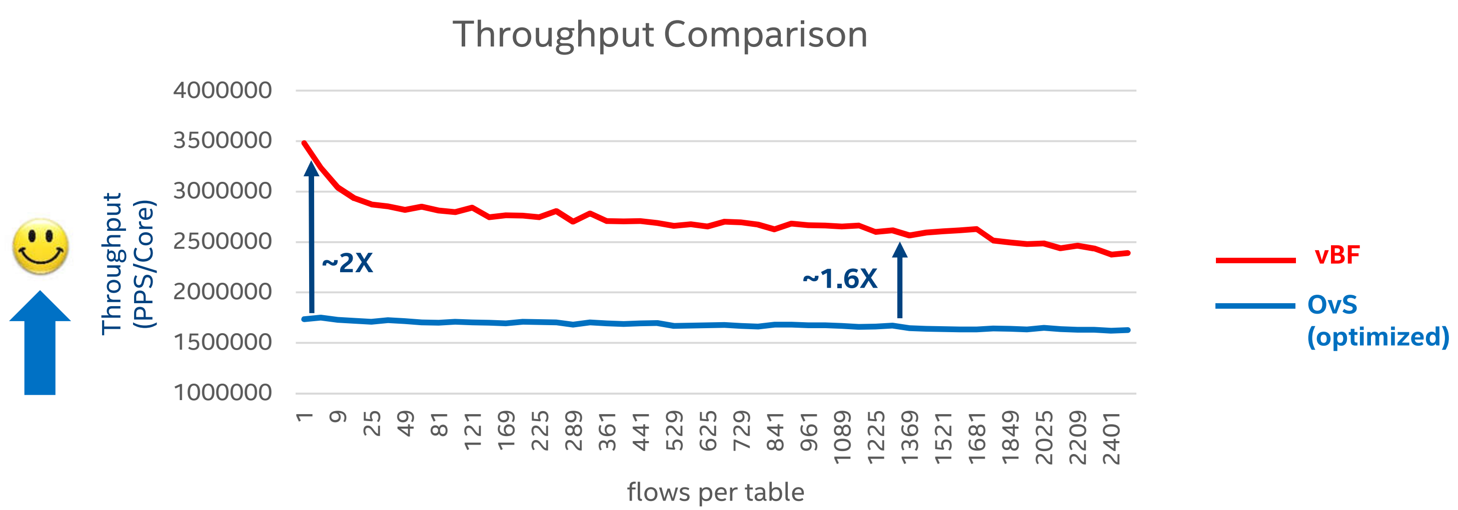


Fig. 20 subtables, and each su-btable contains various numbers of rules. Note that we disabled EMC for showing the benefits.

- vBF provides significant lookup performance gain when compared with native OvS.
- Gain increases with increasing number of sub-tables.

Results and performance figures are for an experimental prototype and is work in progress. The result reflect specific components on a particular test, in specific systems and should not be generalized for actual products. Differences in hardware, software, or configuration will affect actual performance. Results are generated using uniform random traffic with 20 sub-tables running on Intel Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and using OvS 2.6.0..

Conclusion

- Flow Lookup is a performance bottleneck for OvS, especially with increasing number of flows and sub-tables.
- Two layer table architecture optimizes flow lookup in OvS and avoids the sequential search of the sub-tables.
- Vector Bloom Filter (vBF) uses bloom filters as the first layer and can significantly improves lookup performance for OvS.

- Future Work:
  - Investigate other technologies to use as the first layer of indirection.
  - Realistic traffic pattern and workload

Questions?

sameh.gobriel@intel.com
charlie.tai@intel.com