Rethinking Message Buffer Structures for 100 Mpps Cloud-native Network Function

Ayuto Yamada† Ryota Kawashima†
Hiroki Nakayama†† Tsunemasa Hayashi†† Hiroshi Matsuo†

† Nagoya Institute of Technology
†† BOSCO Technologies, Inc.

POINTS

Goal

150 Mpps (≈ 100 Gbps) REAL CNFs for future NW

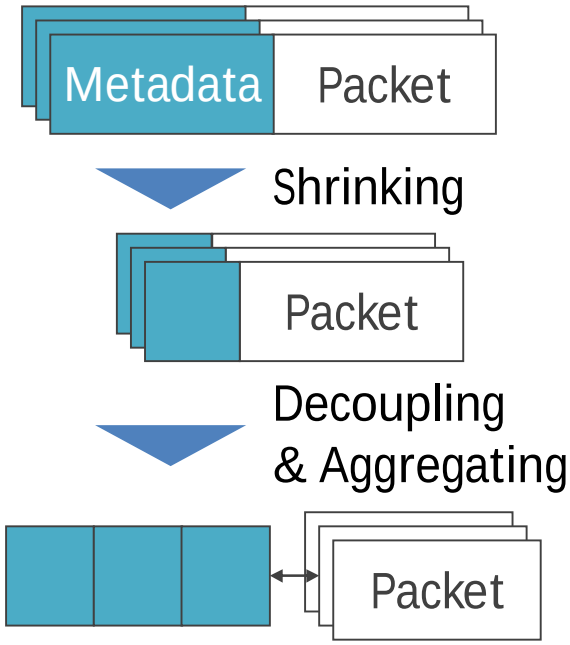
Idea

Redesign Packet Stores to Unlock CPU Potential

Result

Achieved 120+ Mpps from 20 Mpps

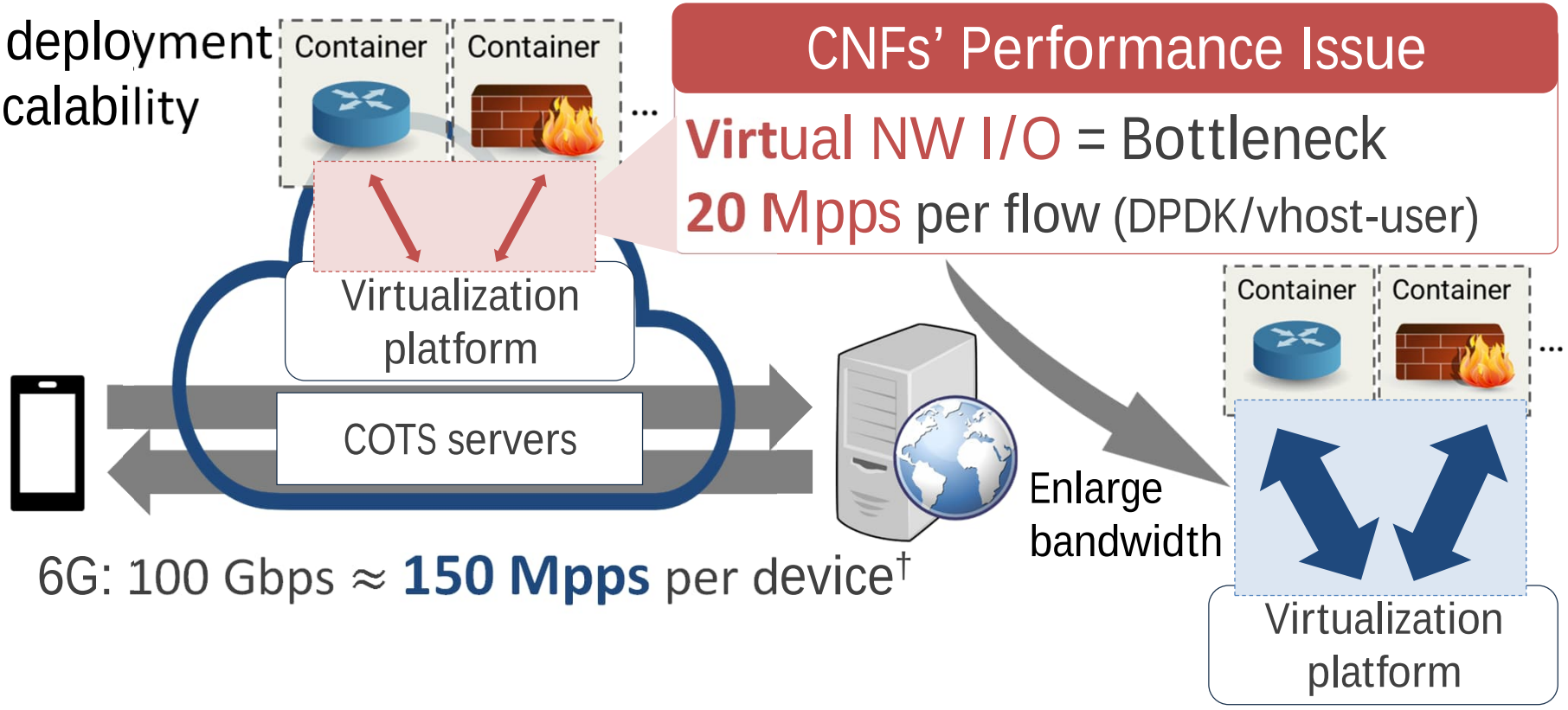
6x higher than DPDK/vhost-user



Fully Virtualized Cloud Native Network

Cloud-native **Network Functions**

- ✓ Rapid deployment
- ✓ High scalability

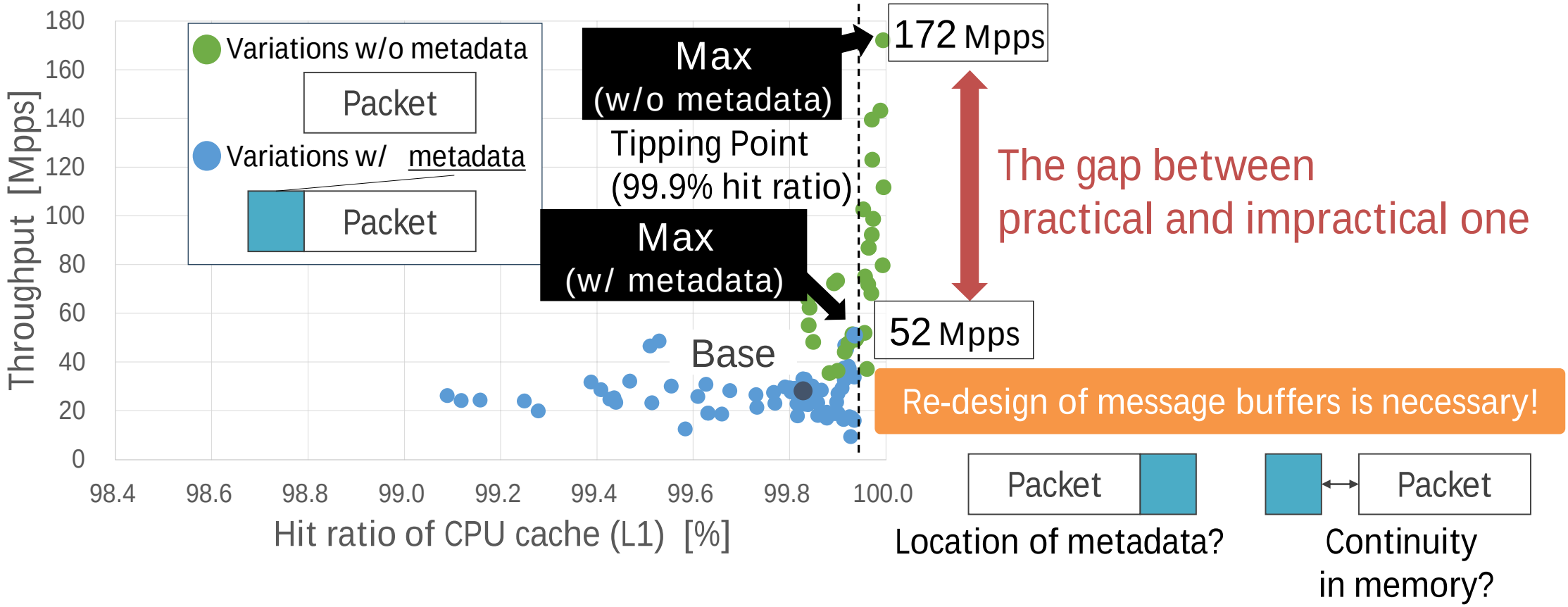


You can use CNFs widely!

† Nokia Bell Labs, "Communications in the 6G era", 2020

Previous Work
- Pursing Efficiency of vNW I/O -

- 99.9% hit ratio of CPU L1 cache was necessary!
- The key performance factor was message buffers!

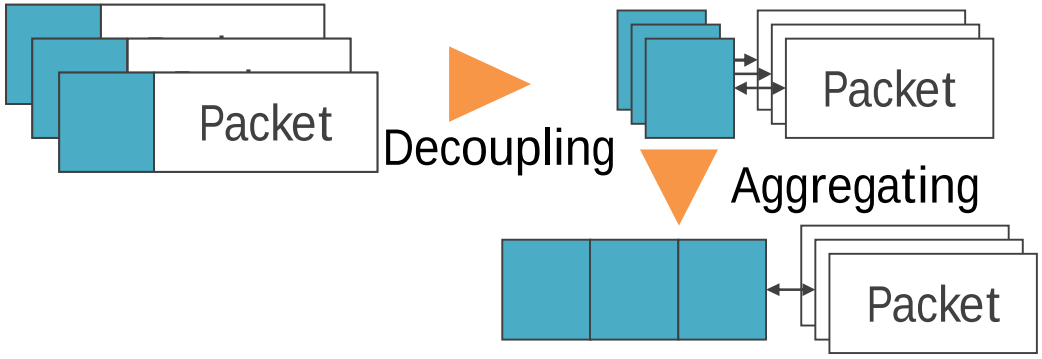


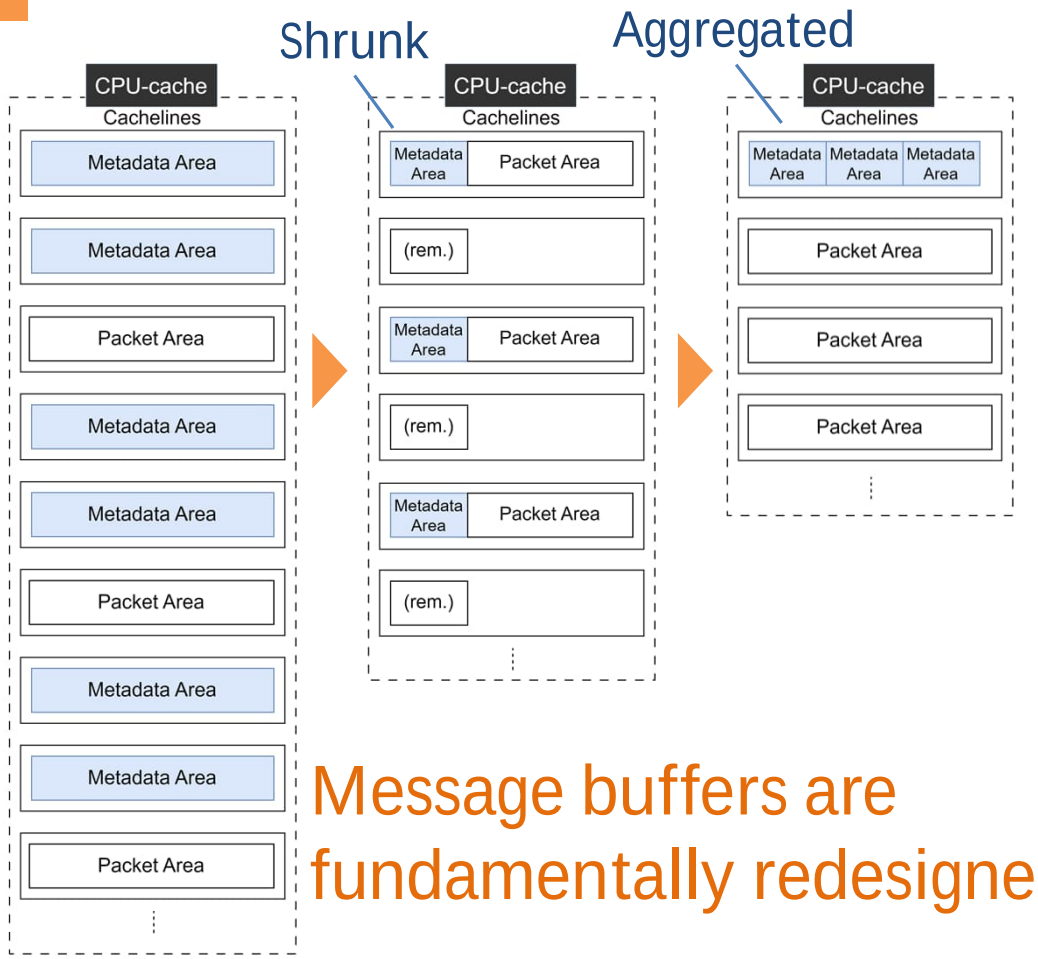
Rethinking Message Buffer Structures

Reducing CPU-cache consumption

Step 1: Shrinking Metadata Area

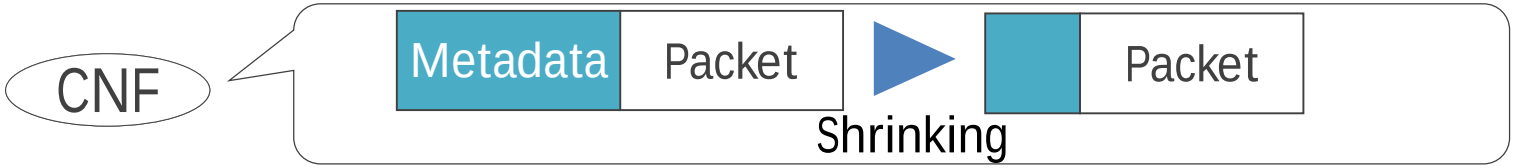
Step 2: Decoupling & Aggregating Metadata Areas



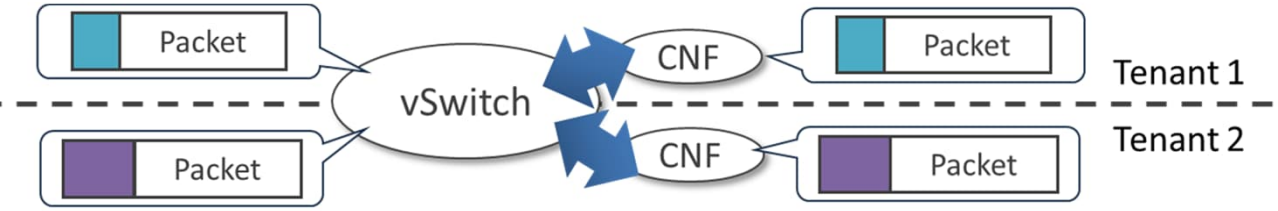


Message buffers are fundamentally redesigned!

Step 1: Shrinking Metadata Area
- Problem Statement -

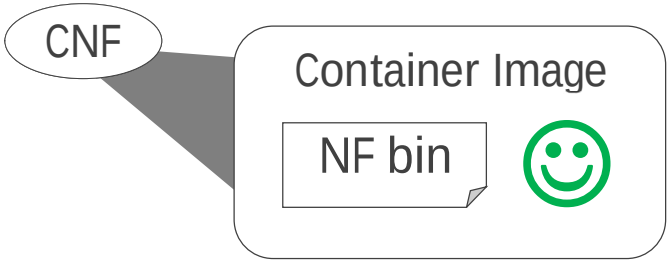


Req. 1: Compatibility with multi-tenant environment



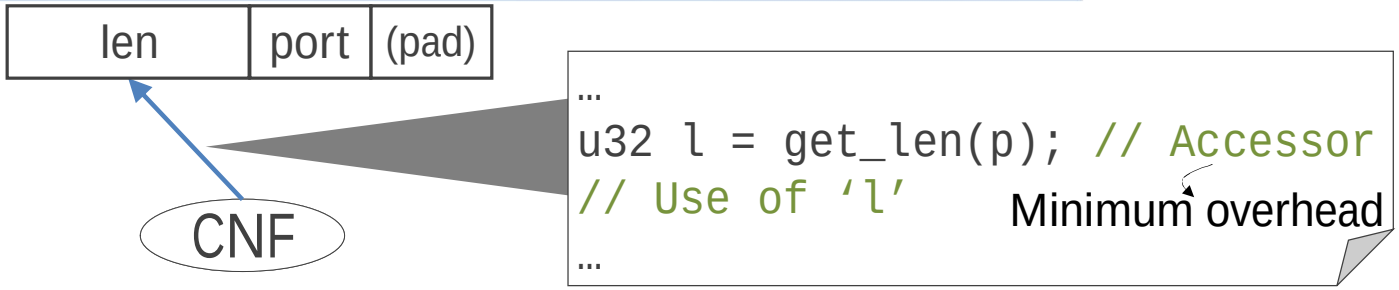
vSwitch coordinates CNFs for each tenant

Req. 2: Keeping usability of CNFs



Avoiding recompilation of NFs

Req. 3: Minimum overhead in performance



Step 1: Shrinking Metadata Area
- Proposed System -

Prepare a Field Definition Table (FDT)

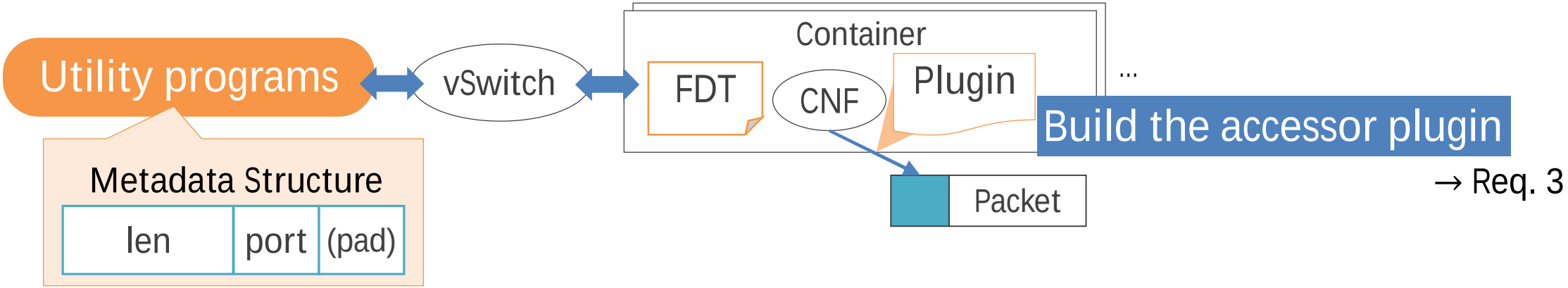
128 B metadata (of DPDK)

len ol_flags
timestamp port
packet_type nb_segs
tx_offload

FDT
port, 2B
len, 4B

Req. 1: Compatibility with multi-tanant environment
Req. 2: Preserving usability of CNFs
Req. 3: Minimum overhead in performance

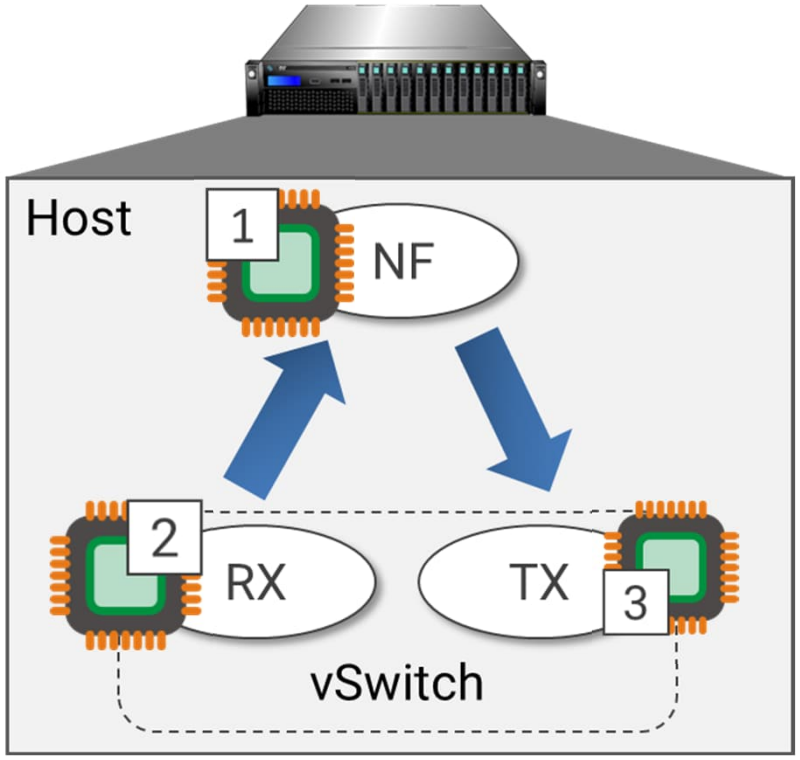
Dynamically compose the metadata structure from FDT → Req. 1 & 2



Build the accessor plugin → Req. 3

Evaluation Environment & Content

A real CPU on server



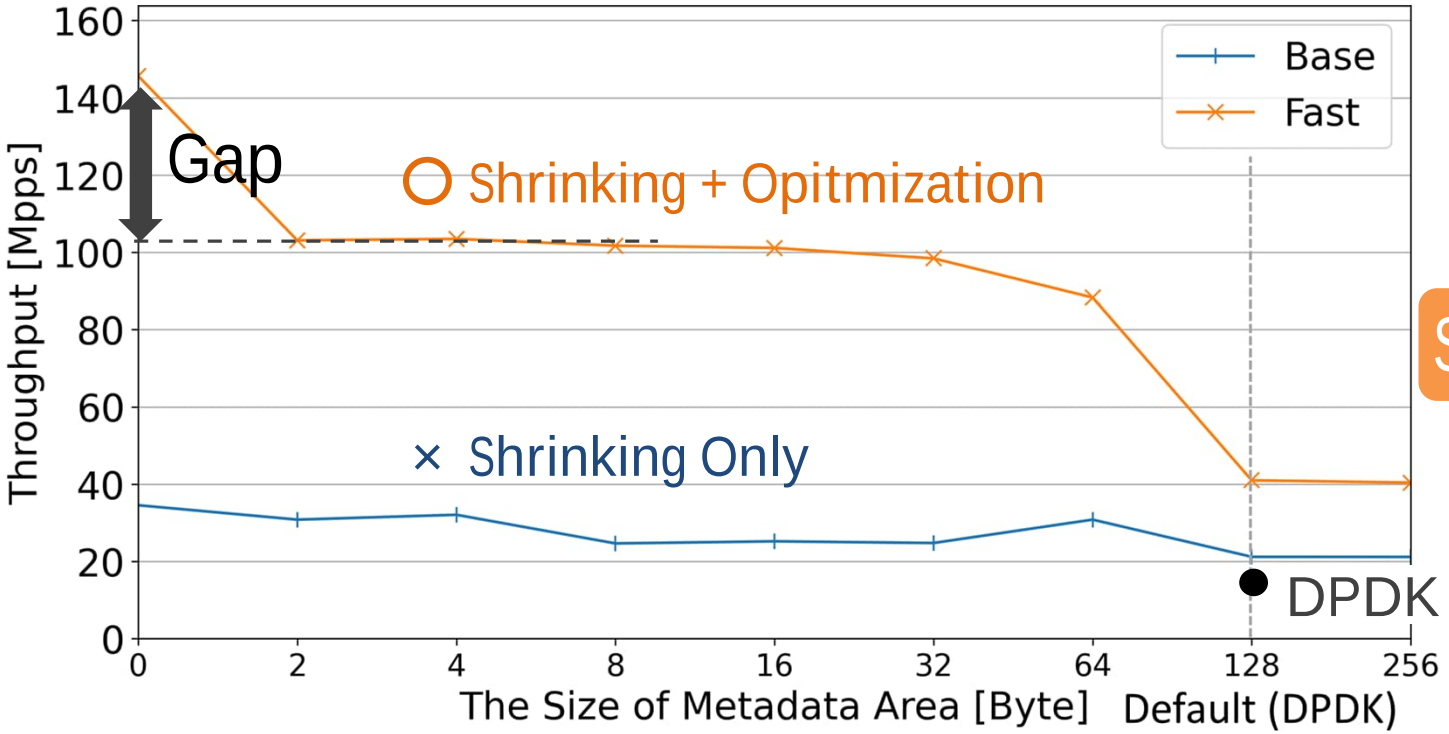
CPU	Core i9 11900K
Clock	3.5 GHz
L1d	48 KiB
L2	0.5 MiB
L3 (shared)	16 MiB
Memory Clock	3200 MHz

Throughput + Cache Usage

- Cache Hit Num.
- Cache Miss Num.
- Replacement Num.
- Read for Ownership
- L1D Fill Buffer Miss

...

Preliminary Evaluation



Shrinking was effective

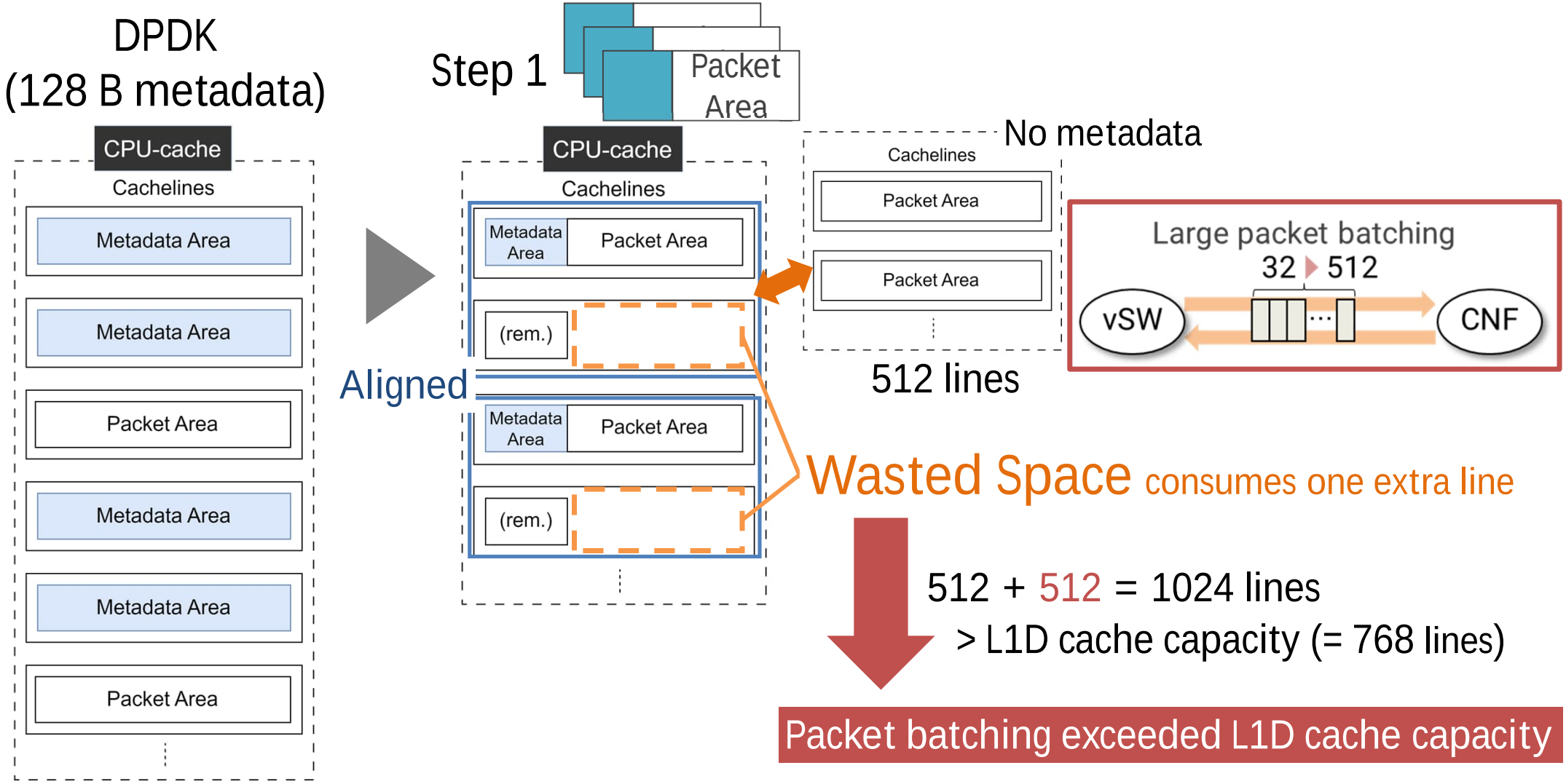
L1D Cache Usage

Metadata size	0	2	4	8	16	32	64
Hit ratio [%]	**99.85**	97.73	97.64	97.75	97.68	97.59	97.64
Replace. [times/packet]	**1.79**	5.57	5.64	5.58	5.66	5.74	6.10

→ Under tipping point

Replace. is the key factor?

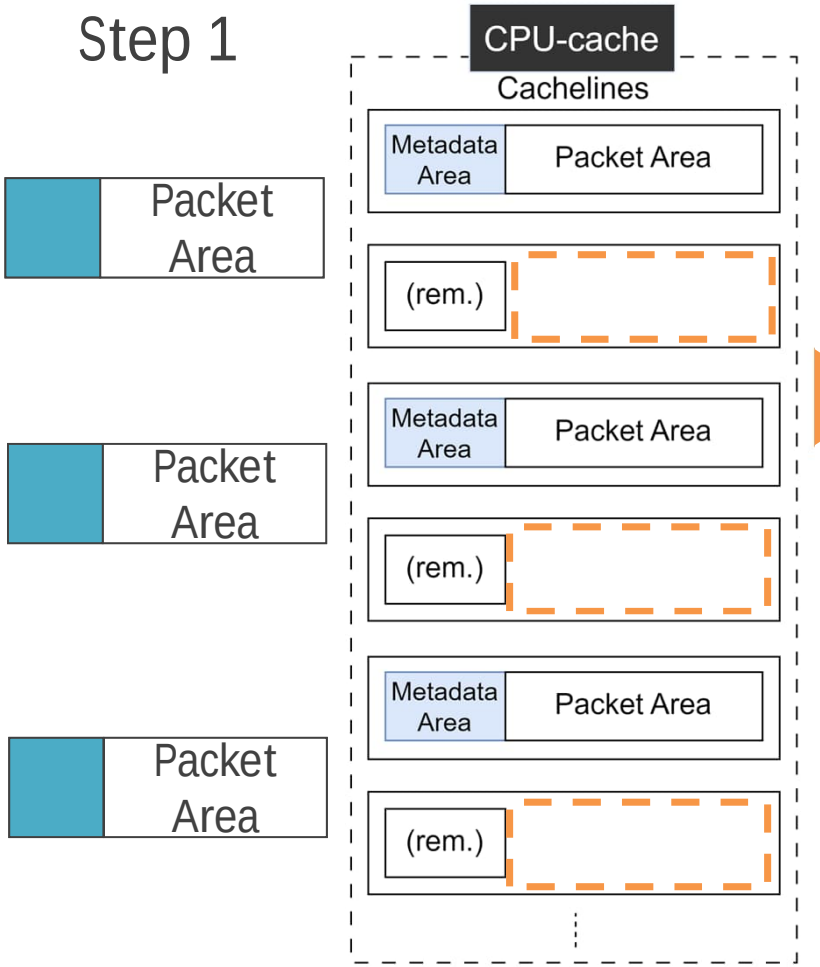
Step 2: Decoupling & Aggregating Metadata Areas
- Problem Statement -



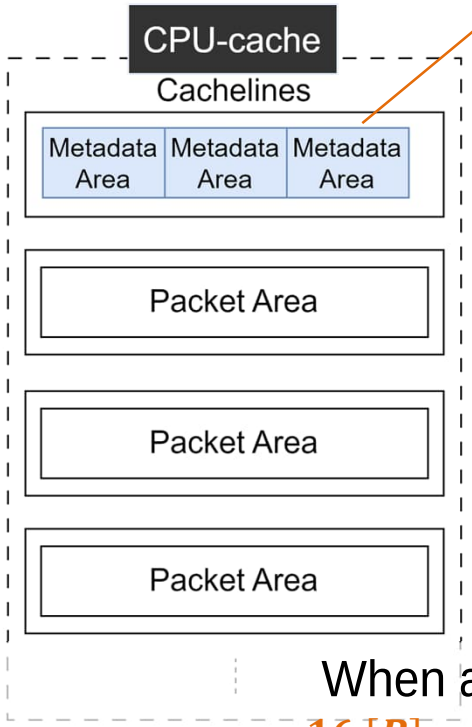
Step 2: Decoupling & Aggregating Metadata Areas
- Proposed Message Buffers -

Leaves out wasted space

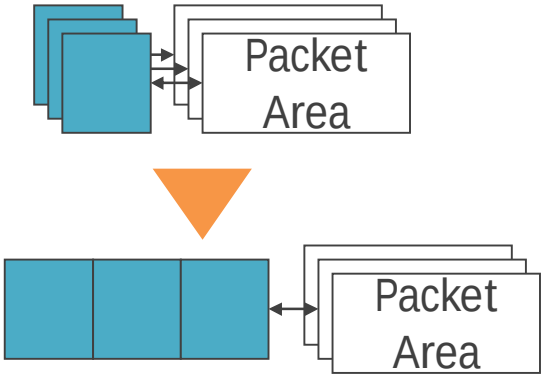
Step 1



Step 1 & Step 2



Aggregates metadata areas into one cacheline



Decoupling & Aggregating

When a metadata area is 16 B,

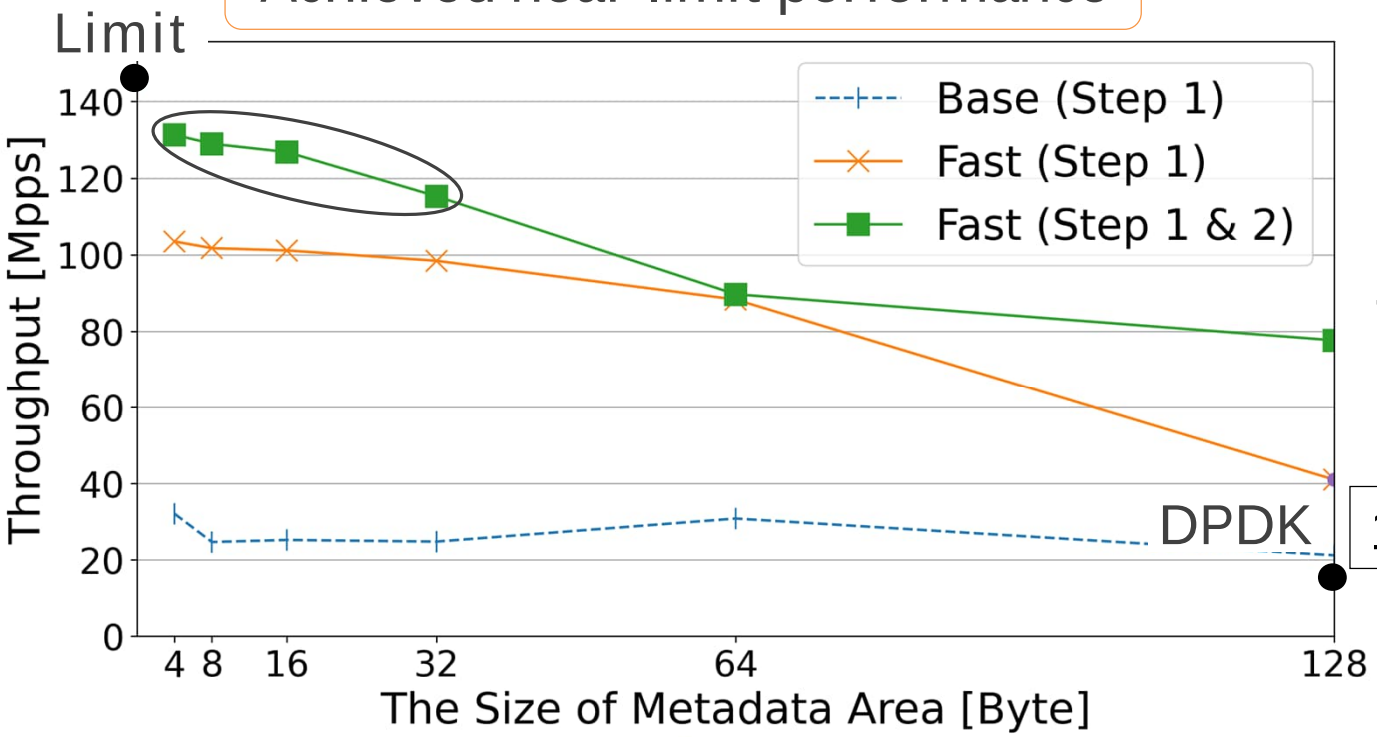
$$512 + 512 \times \frac{16\,[B]}{64\,[B]} = 640 \text{ lines} < \text{the capacity (768 lines)}$$

64 [B] : Cacheline Size

Reducing the num. of used cachelines

Proposal's Effect

Achieved near-limit performance



~120+ Mpps (when 32 B or below)

6x higher than DPDK/vhost-user

17 Mpps

Improved L1D Cache Usage!

		Fast (Step 1 & 2)			
Metadata size	0 B	4 B	8 B	16 B	64 B
Hit ratio	99.85%	99.84%	99.84%	99.85%	98.56%
Replace.[times/packet]	1.79	1.80	1.83	1.83	3.72

Discussion

Why is the throughput lower than the limit (≈ 150 Mpps)?

Efficiency of L1D cache usage was equivalent

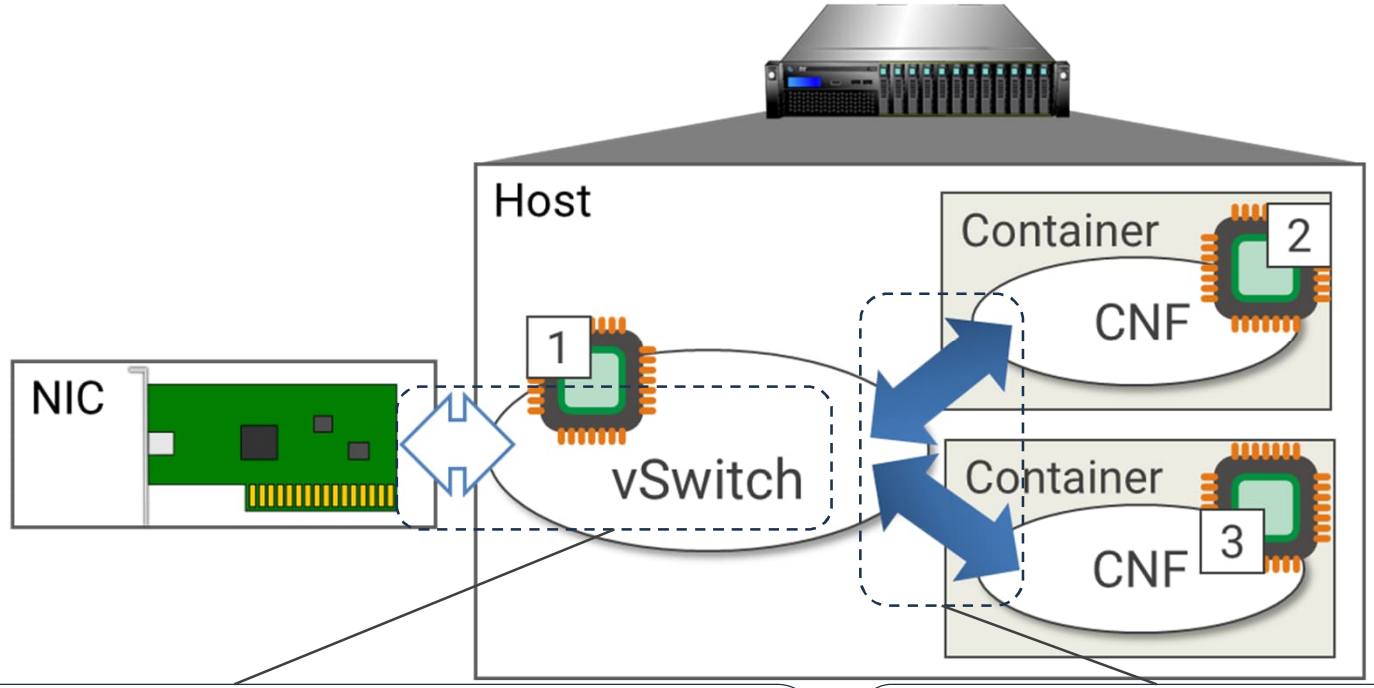
	Throughput [Mpps]	Hit ratio	Replace. [times / packet]
Result (w/ metadata)	131.30	99.84%	1.80
Limit (w/o metadata)	145.63	99.85%	1.79

Reason: Total latency of cache access to metadata

→ 4 ns / packet cf. 6.67 ns / packet (processing time when 150 Mpps)

Measures: All-at-once access (e.g., vectorization)

Related Work



Physical NW I/O

- Optimize the usage of Last-Level Cache[18]
- Simplify the management of message buffers[20]

Ignore virtual NW I/O and L1D cache usage

Virtual NW I/O

Eliminate packet copying

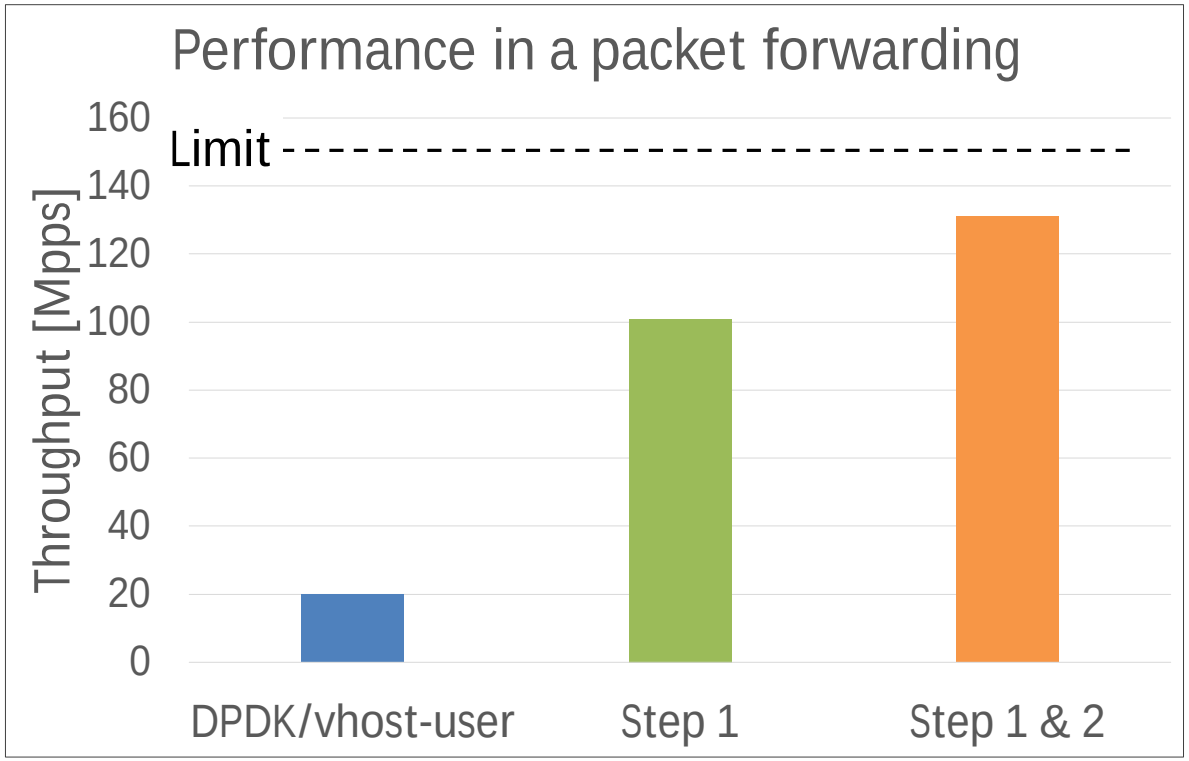
This alone has little effect (~ 30 Mpps)

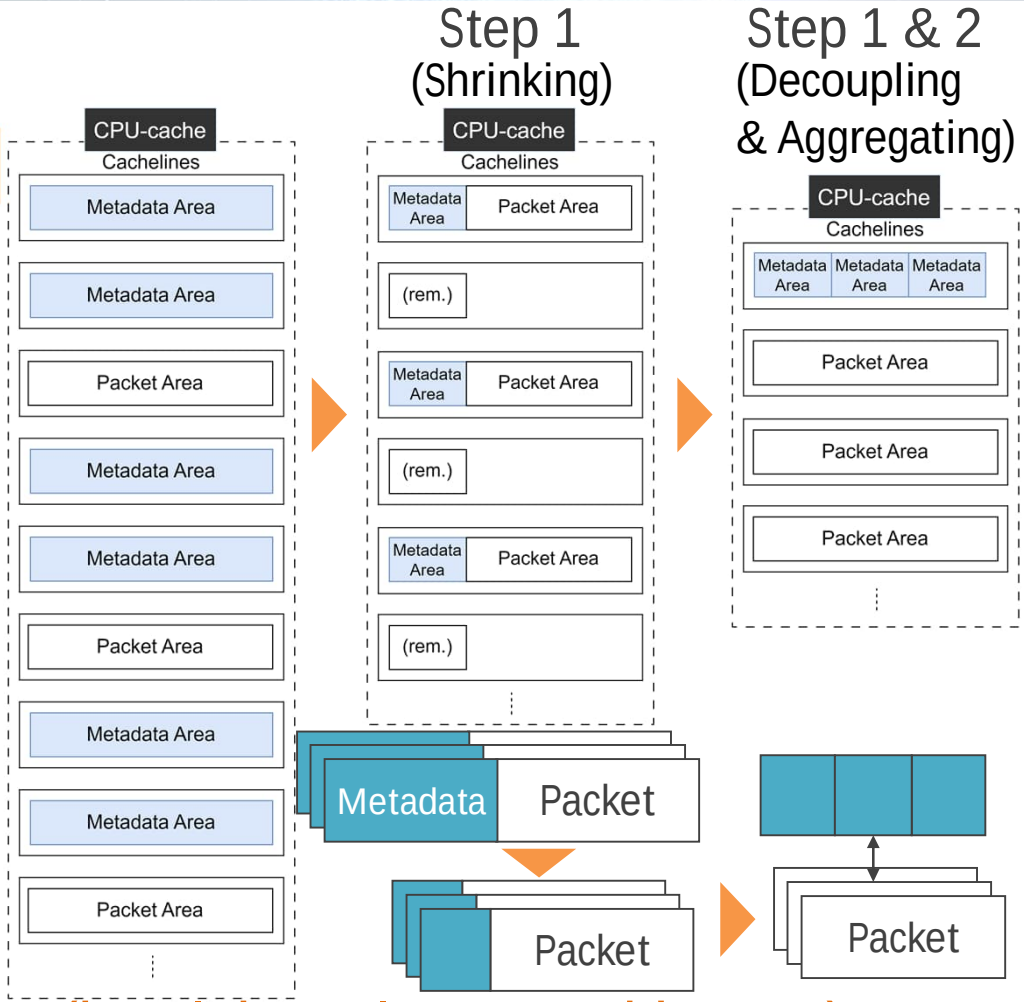
The combination with our proposal maximizes performance

Conclusion

Problem: Not enough performance for future NW

Redesign packet stores to save CPU-cache blocks





Future Work: Try more practical CNFs (involving a large working set)