PARALLEL HARDWARE OFFLOADS

OVSCON 2021 - GAETAN RIVET

AGENDA

Hardware offloads in userland OVS

Parallel offload architecture

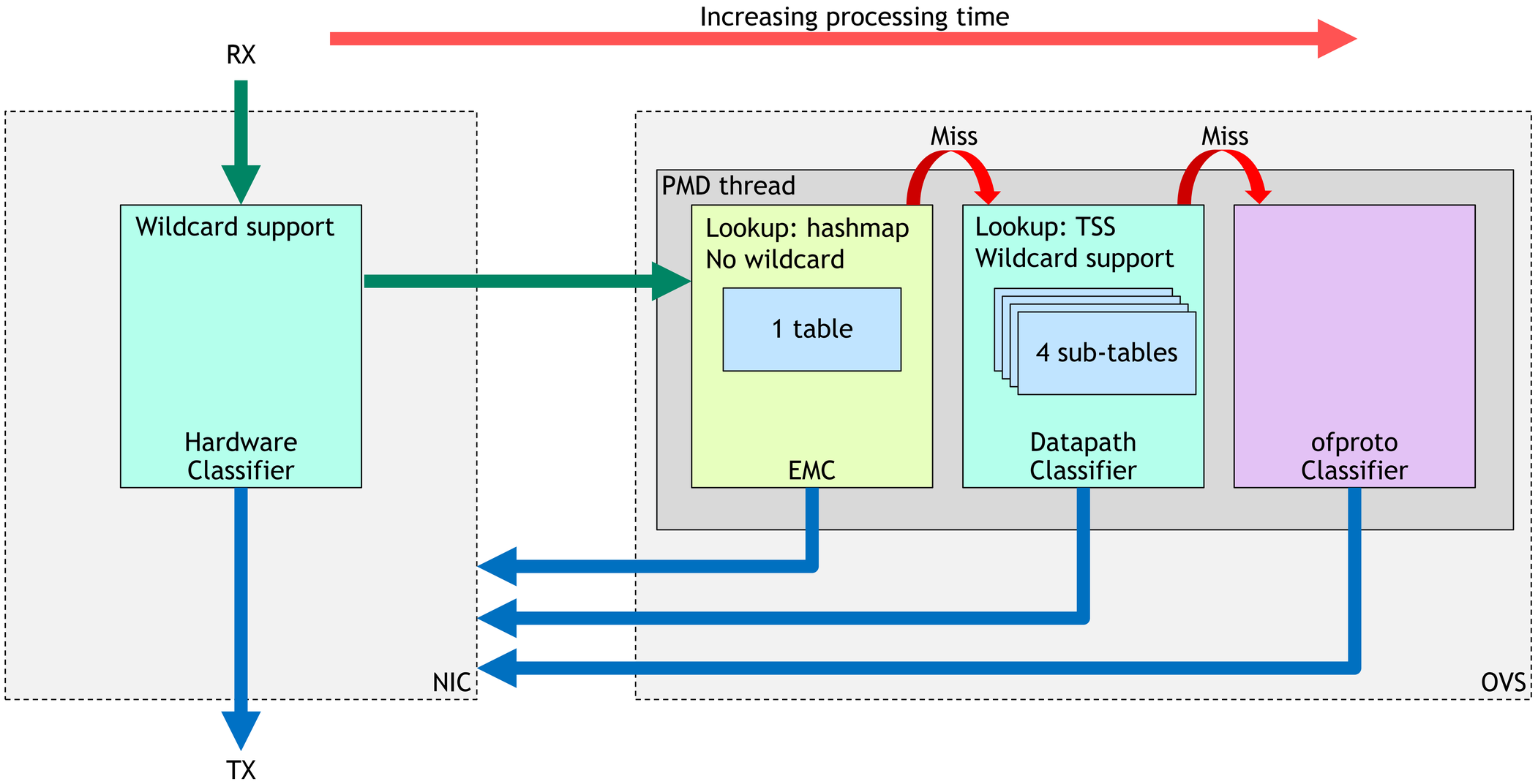
Implementation notes

Results

HARDWARE OFFLOADS IN USERLAND OVS

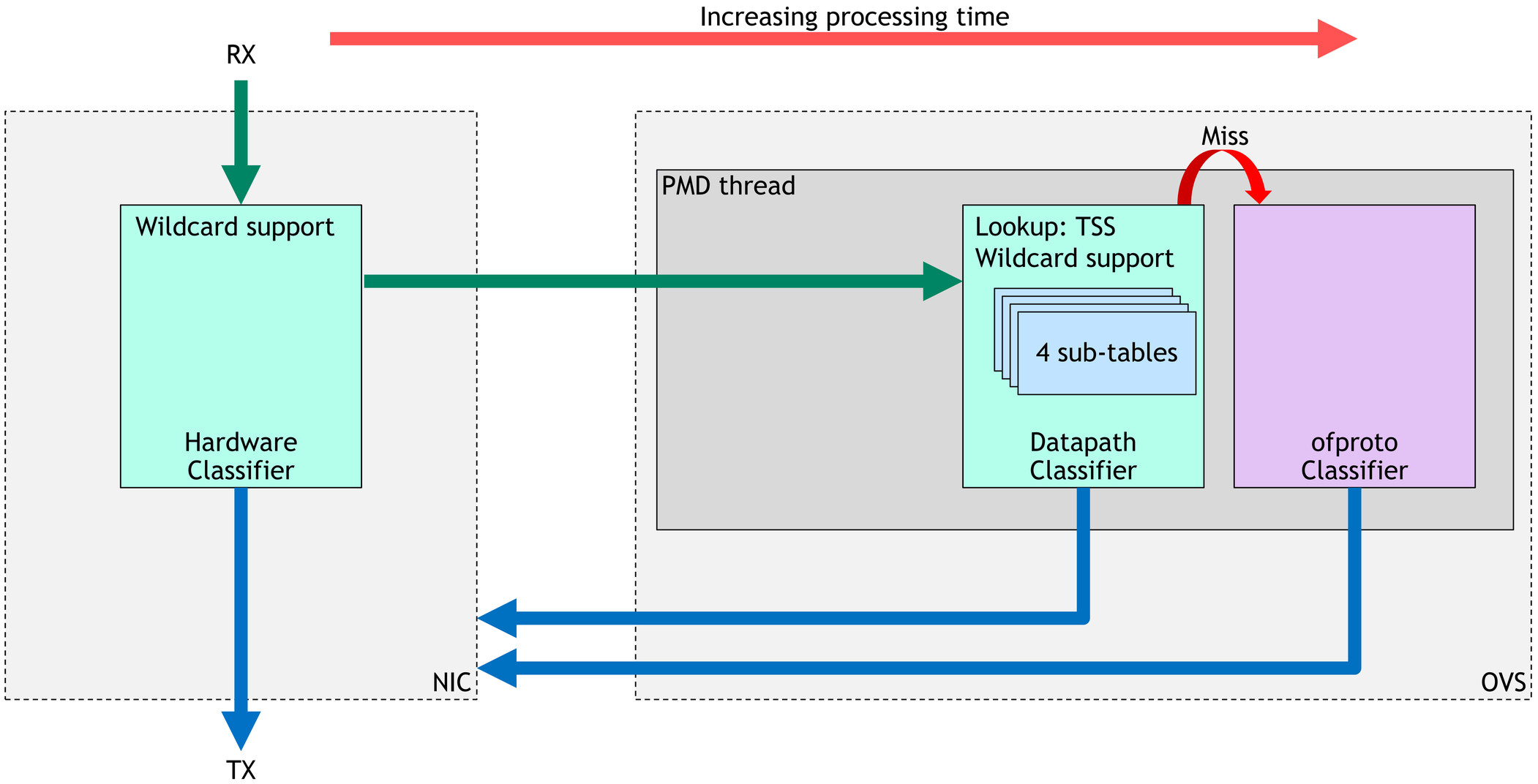
HARDWARE OFFLOADS IN USERLAND OVS

OVS: multi-layer switch



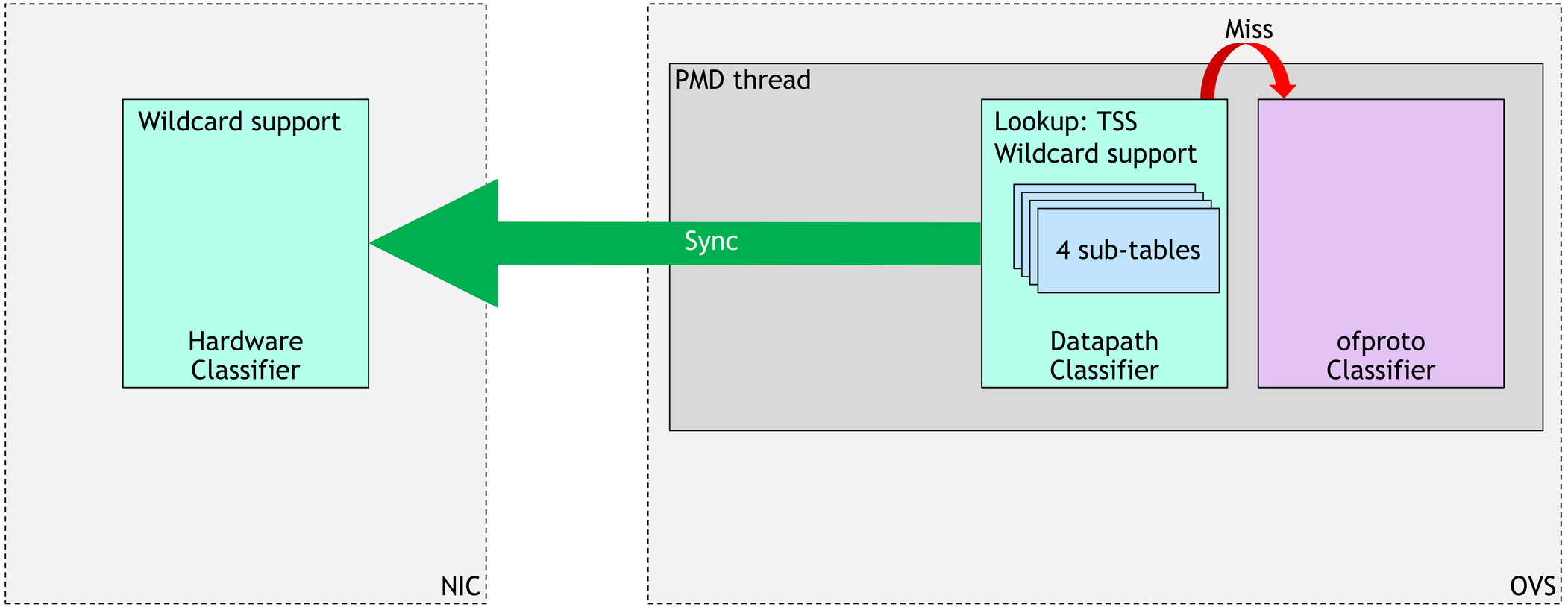
HARDWARE OFFLOADS IN USERLAND OVS

OVS: multi-layer switch



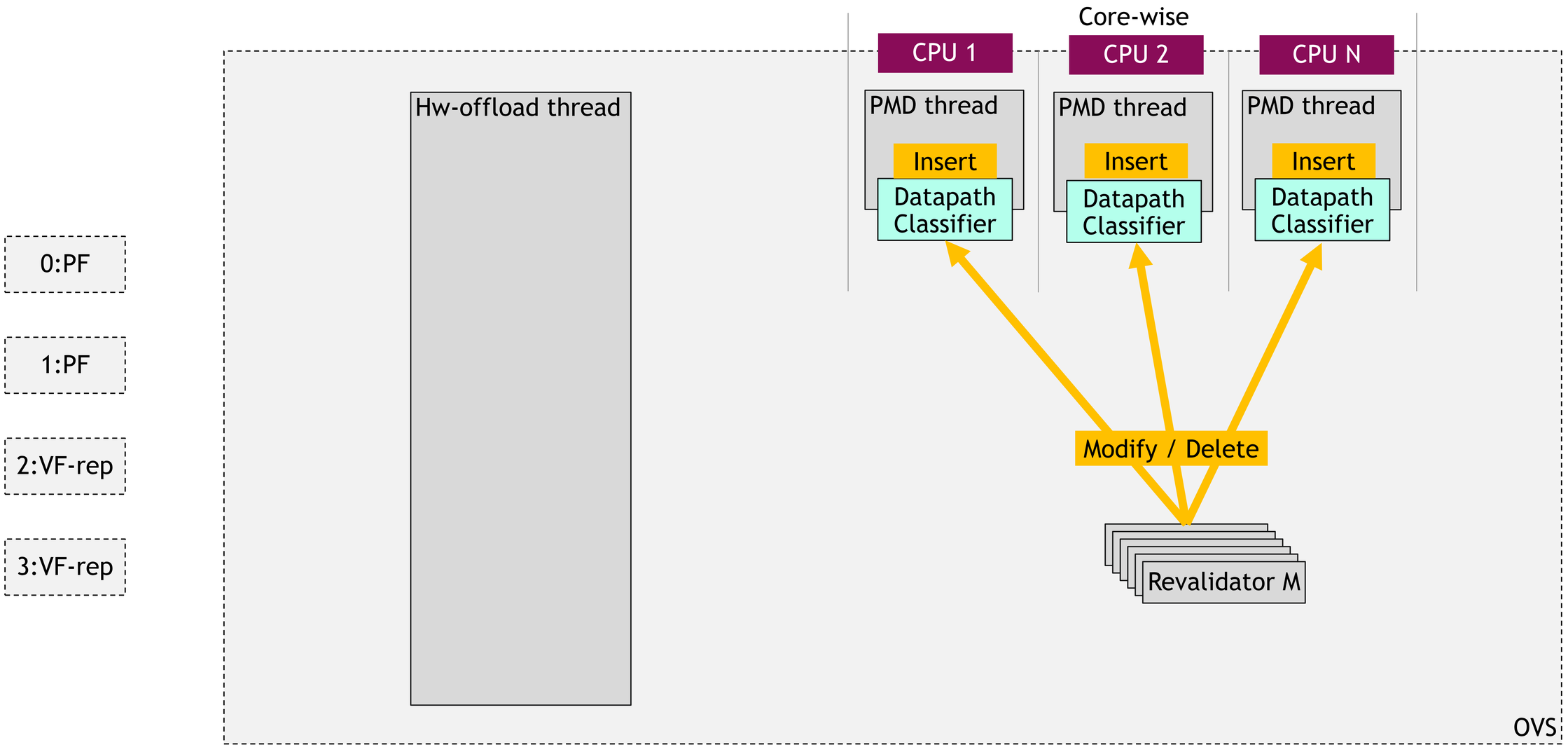
HARDWARE OFFLOADS IN USERLAND OVS

OVS: multi-layer switch



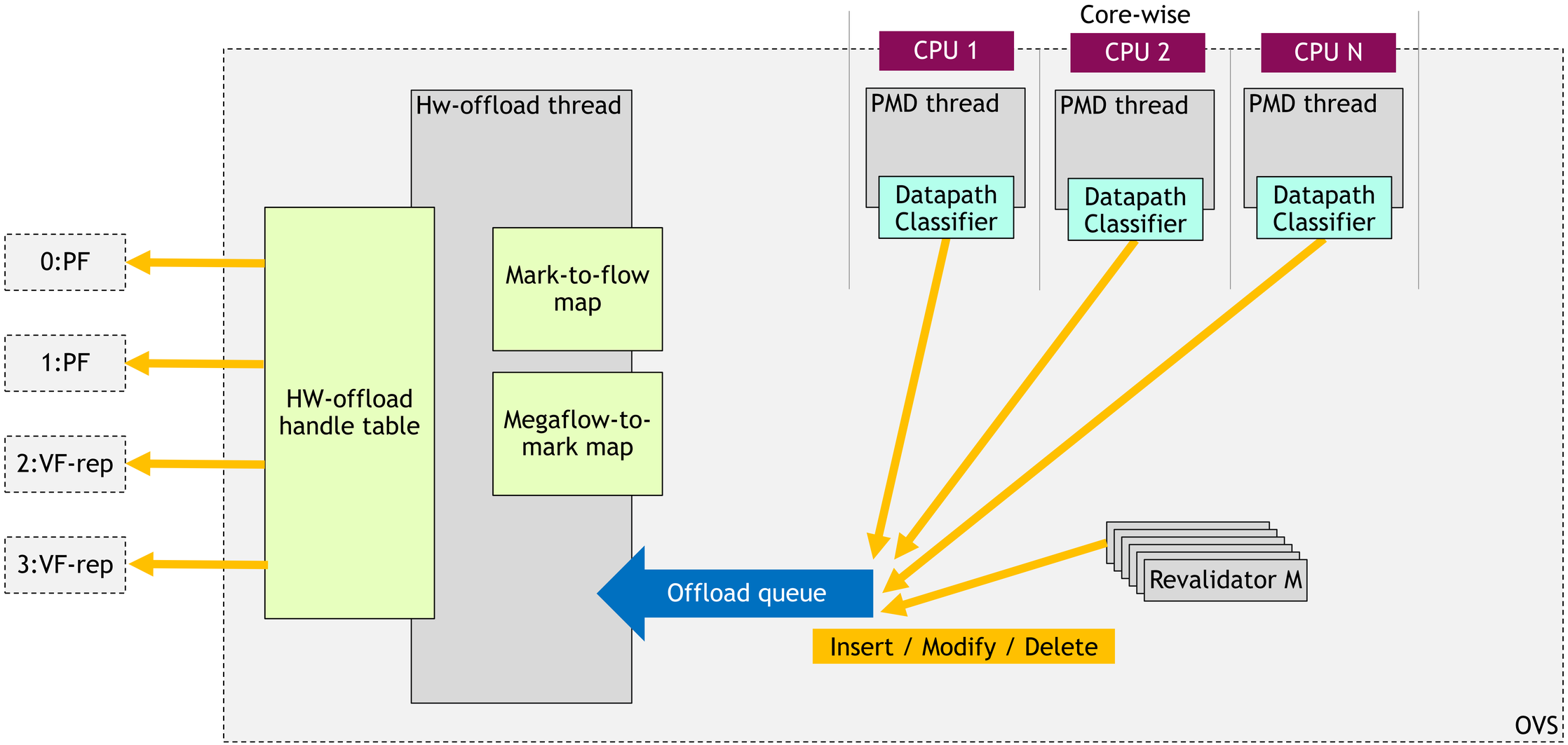
HARDWARE OFFLOADS IN USERLAND OVS

Thread model: DPDK ports



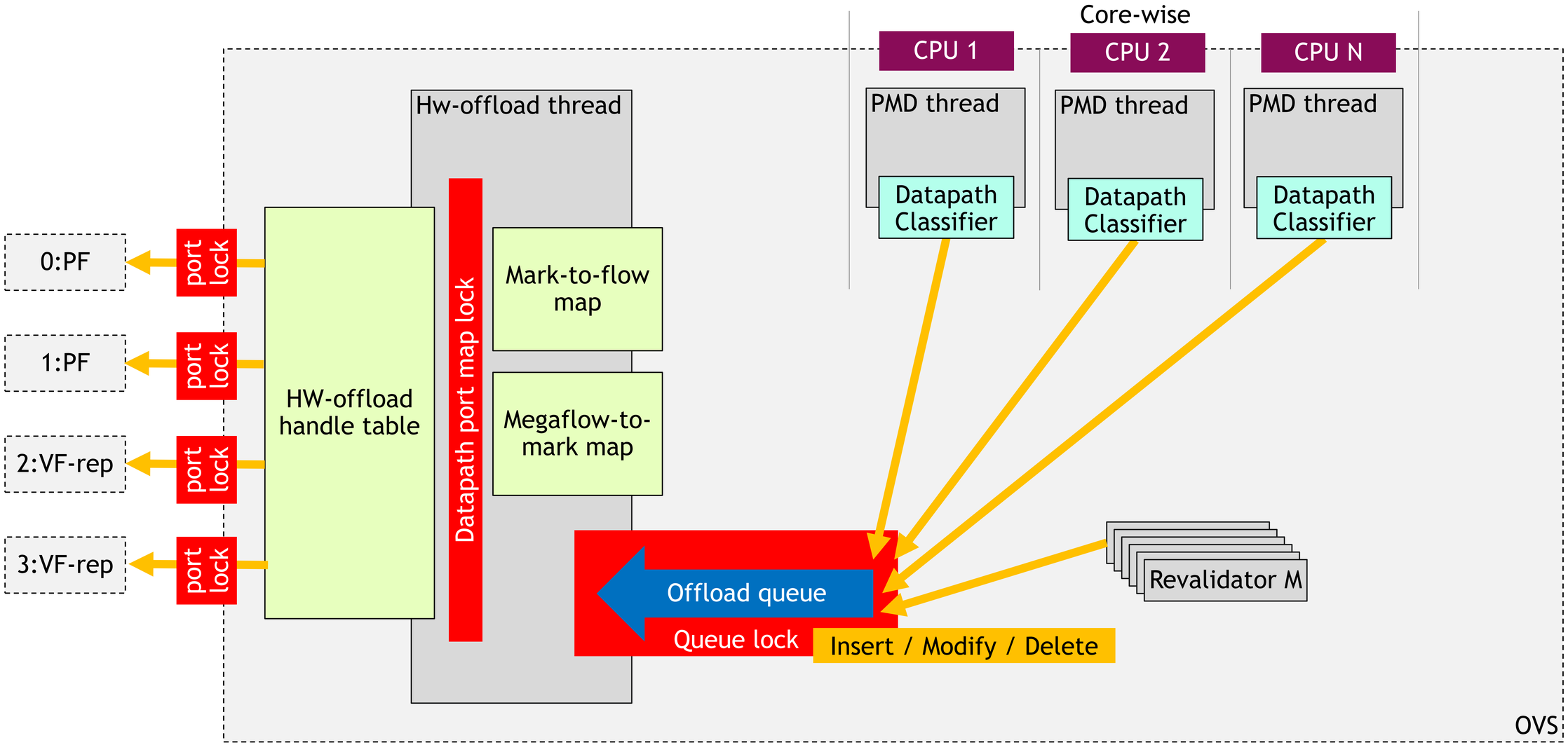
HARDWARE OFFLOADS IN USERLAND OVS

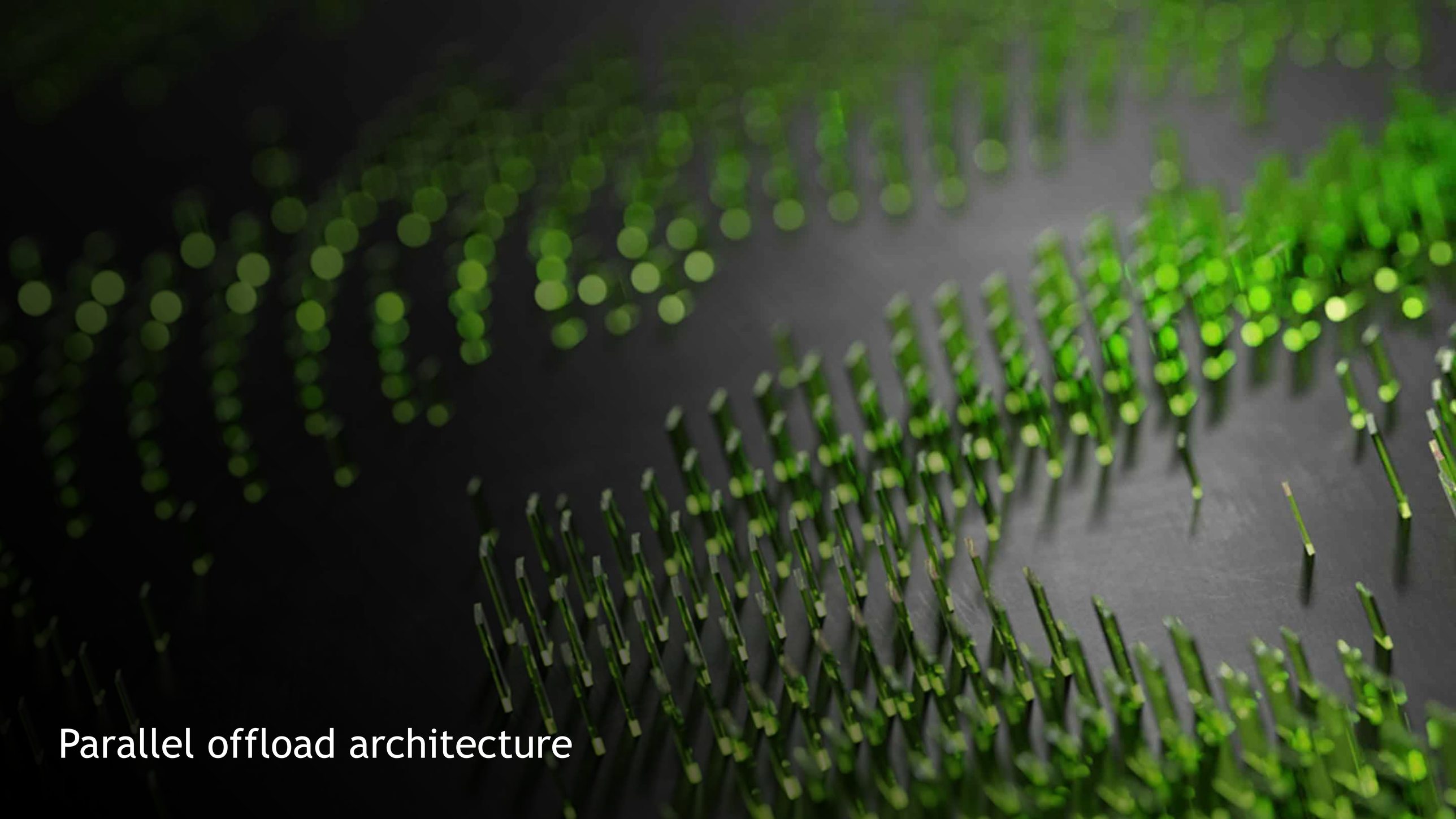
Thread model: DPDK ports



HARDWARE OFFLOADS IN USERLAND OVS

Thread model: DPDK ports

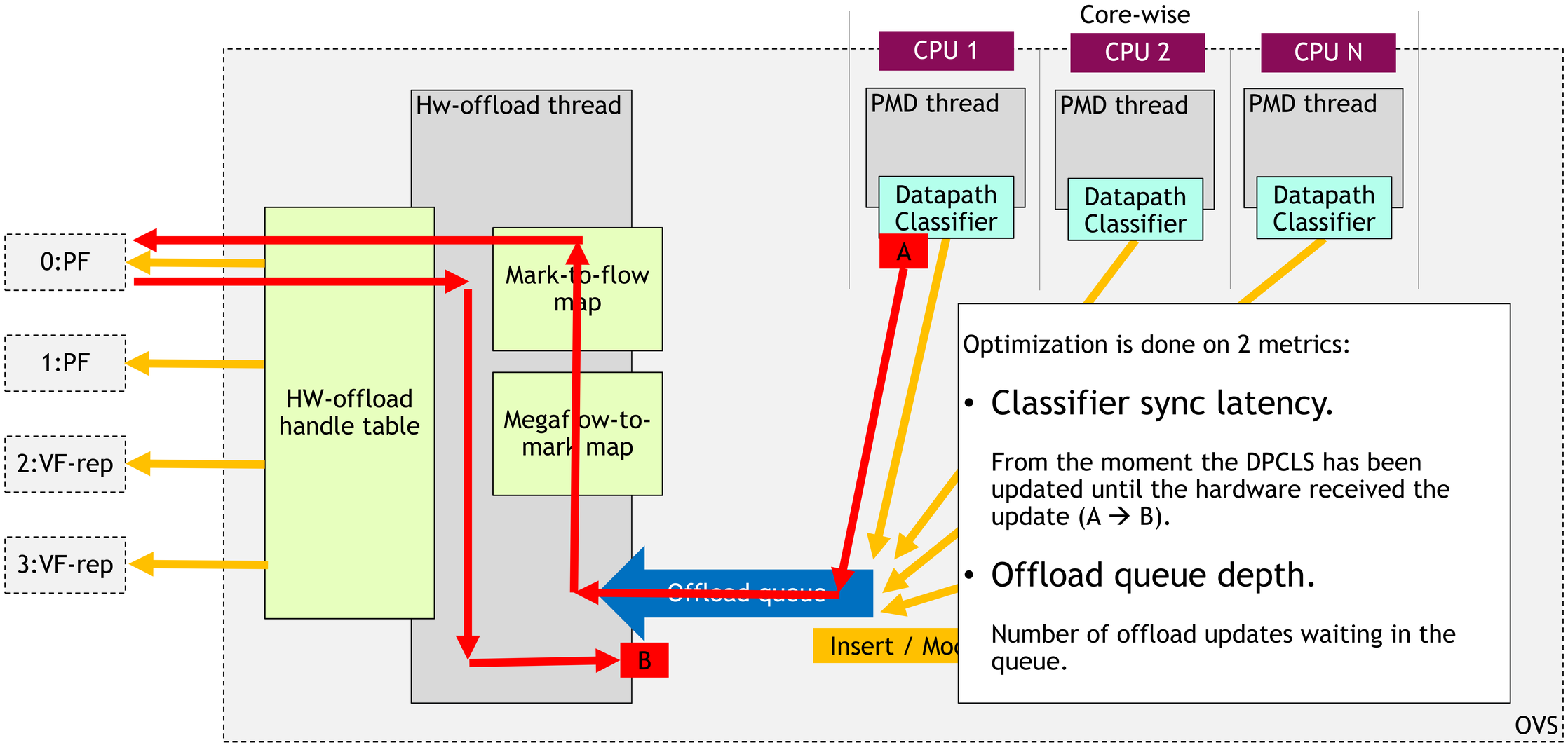




Parallel offload architecture

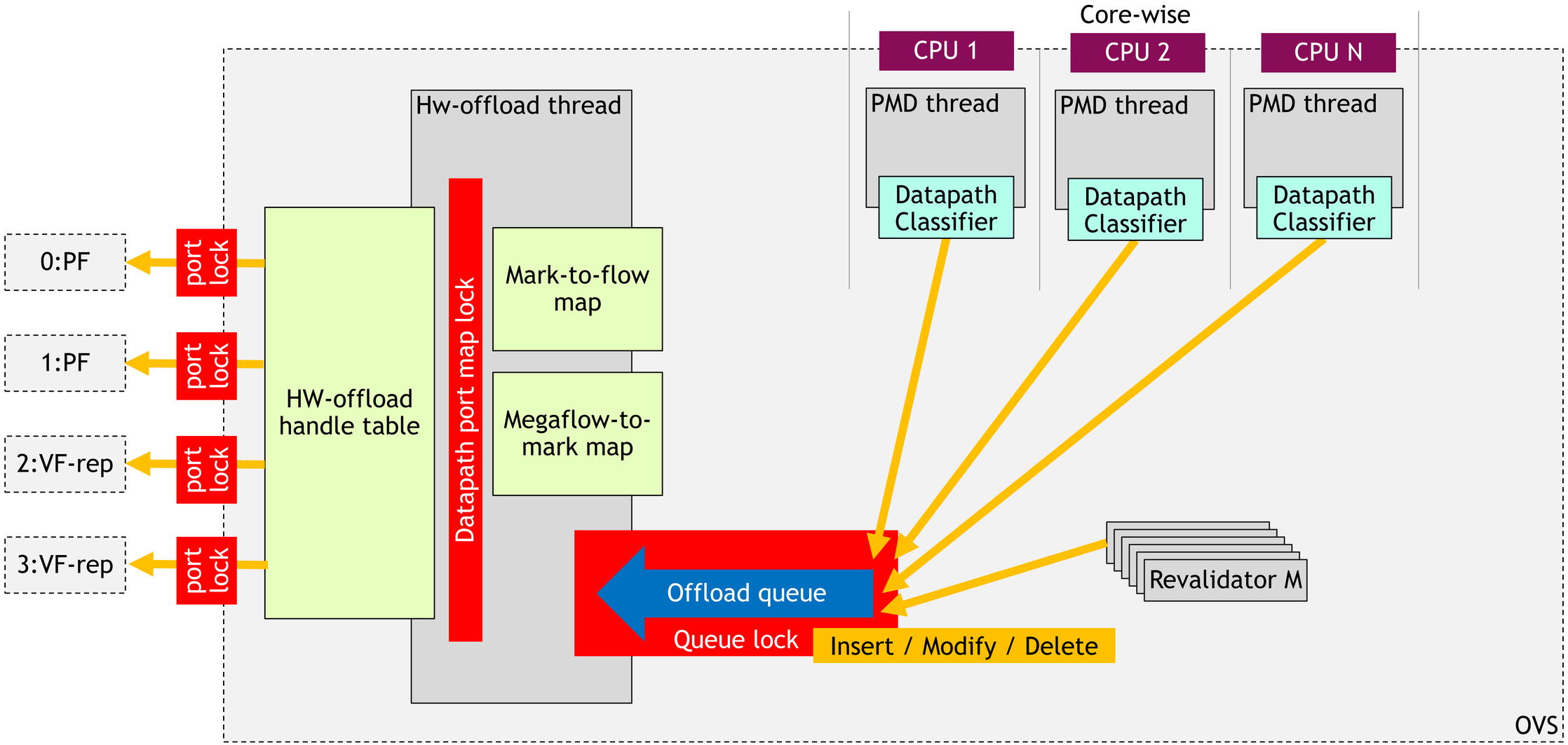
PARALLEL OFFLOAD ARCHITECTURE

Relevant metrics



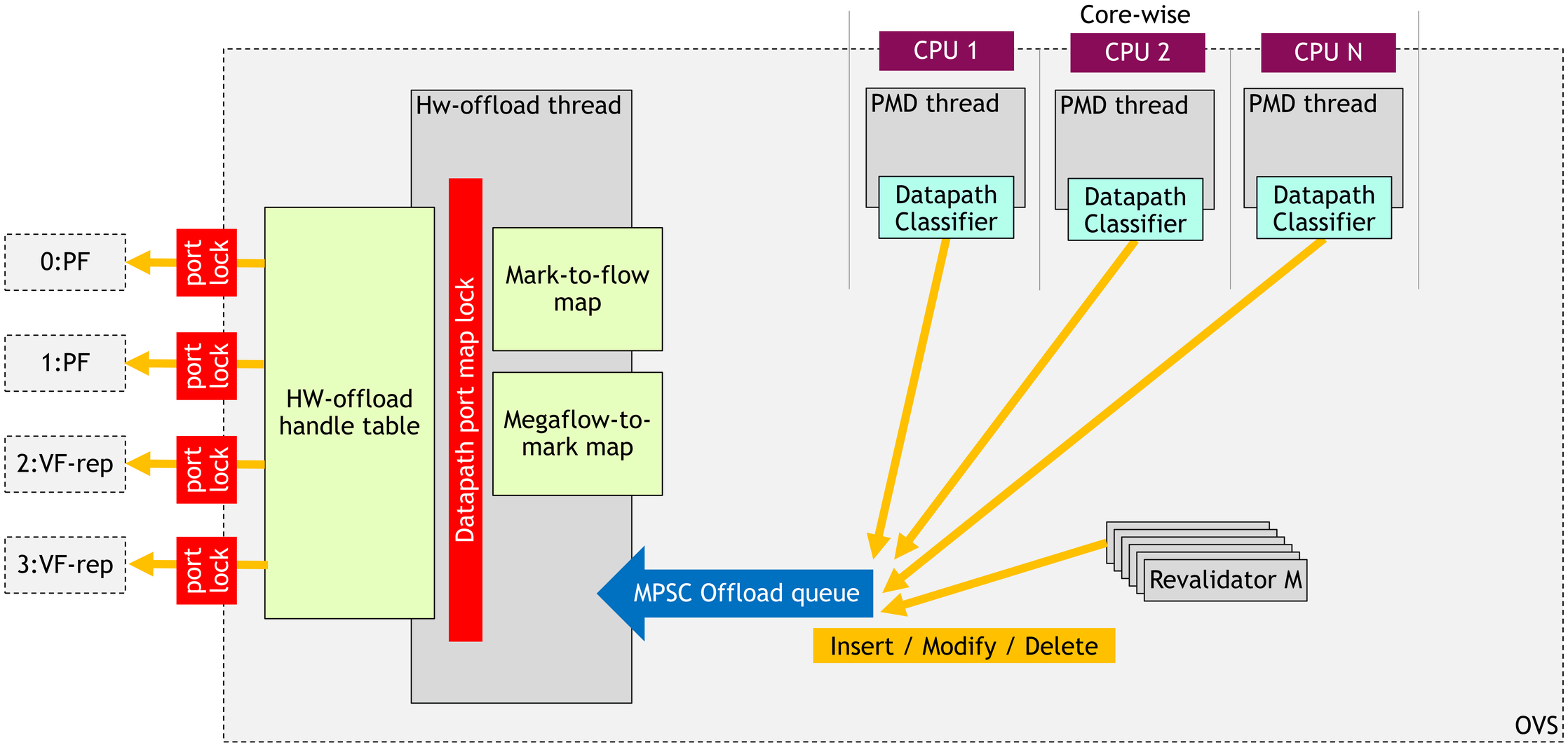
PARALLEL OFFLOAD ARCHITECTURE

Current state



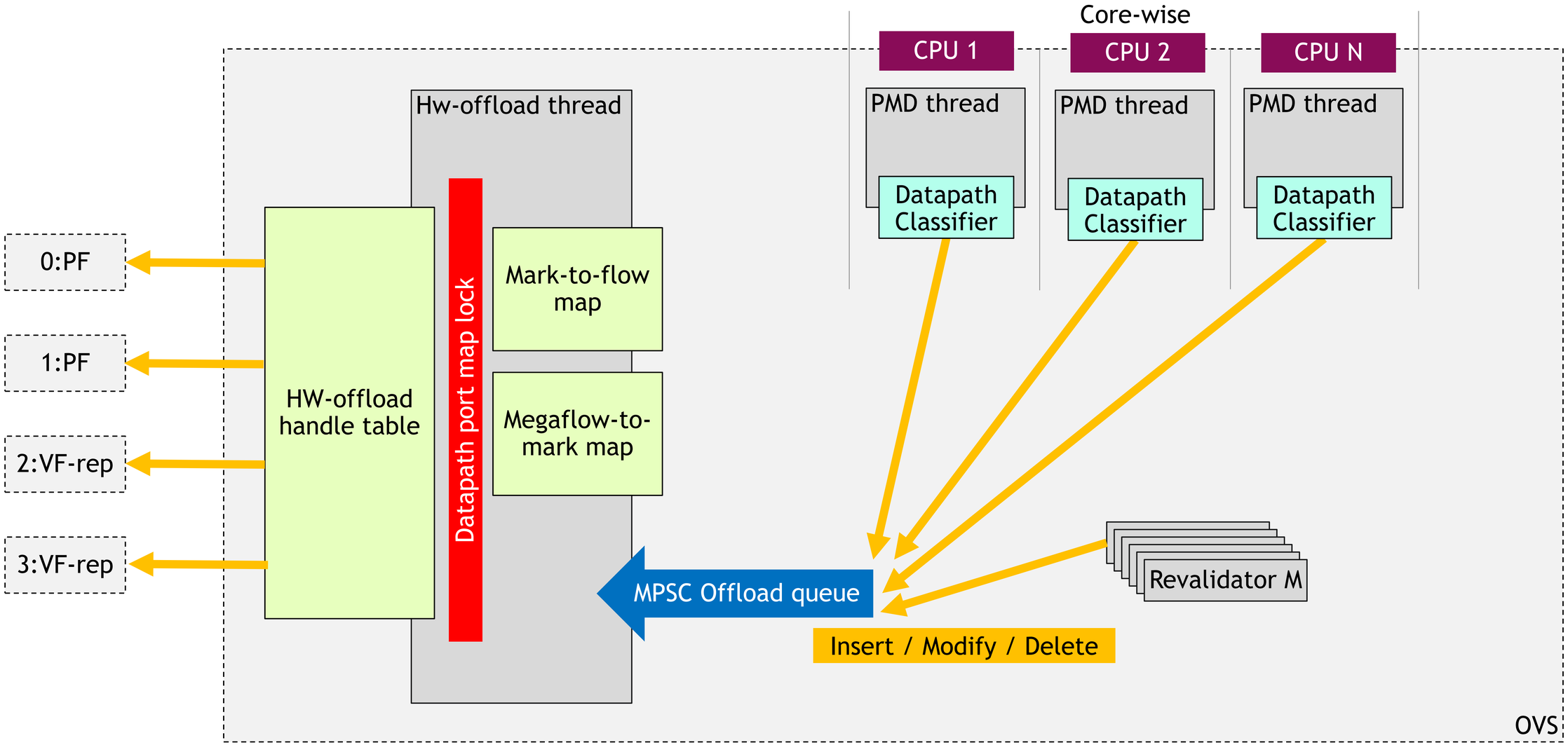
PARALLEL OFFLOAD ARCHITECTURE

Locking changes 1



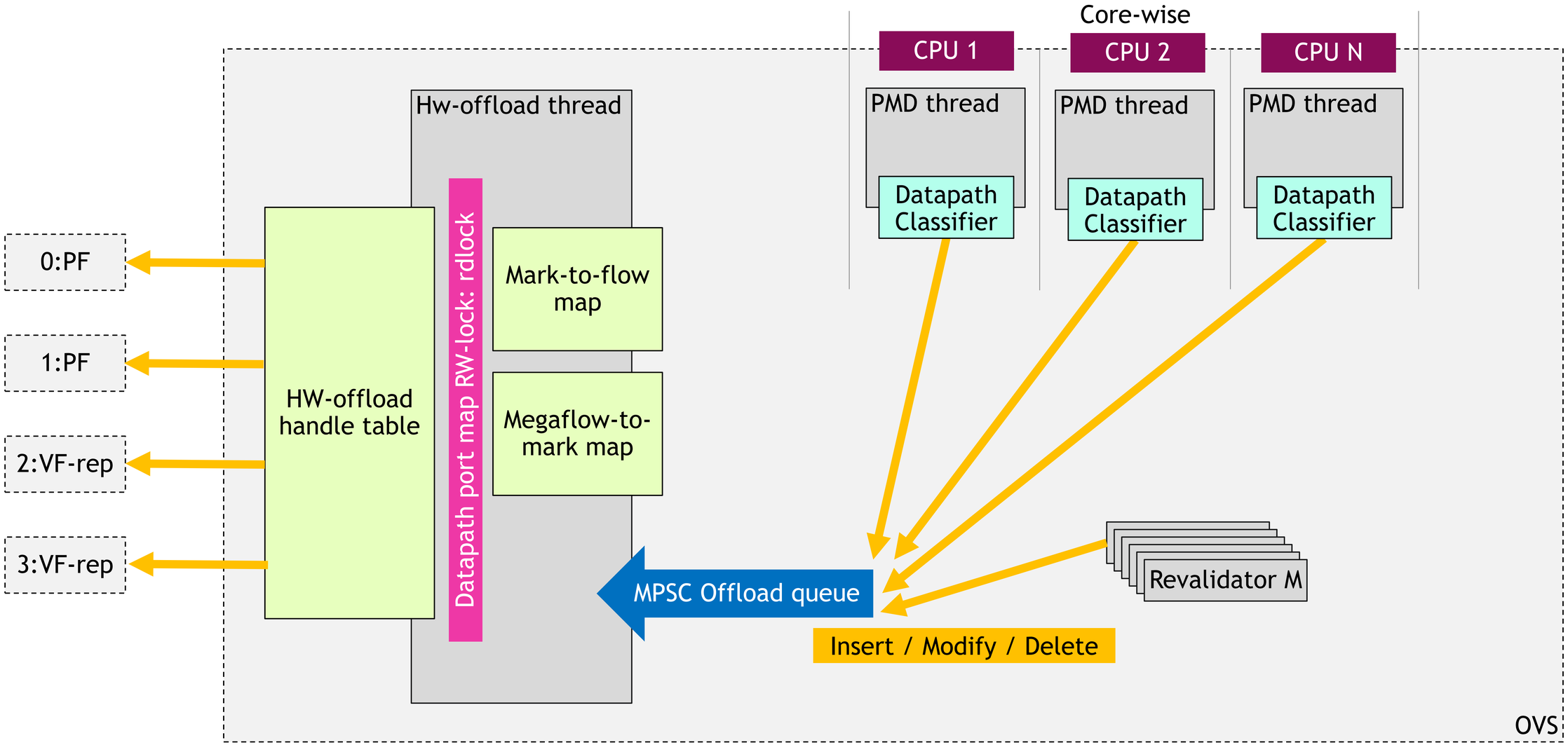
PARALLEL OFFLOAD ARCHITECTURE

Locking changes 2



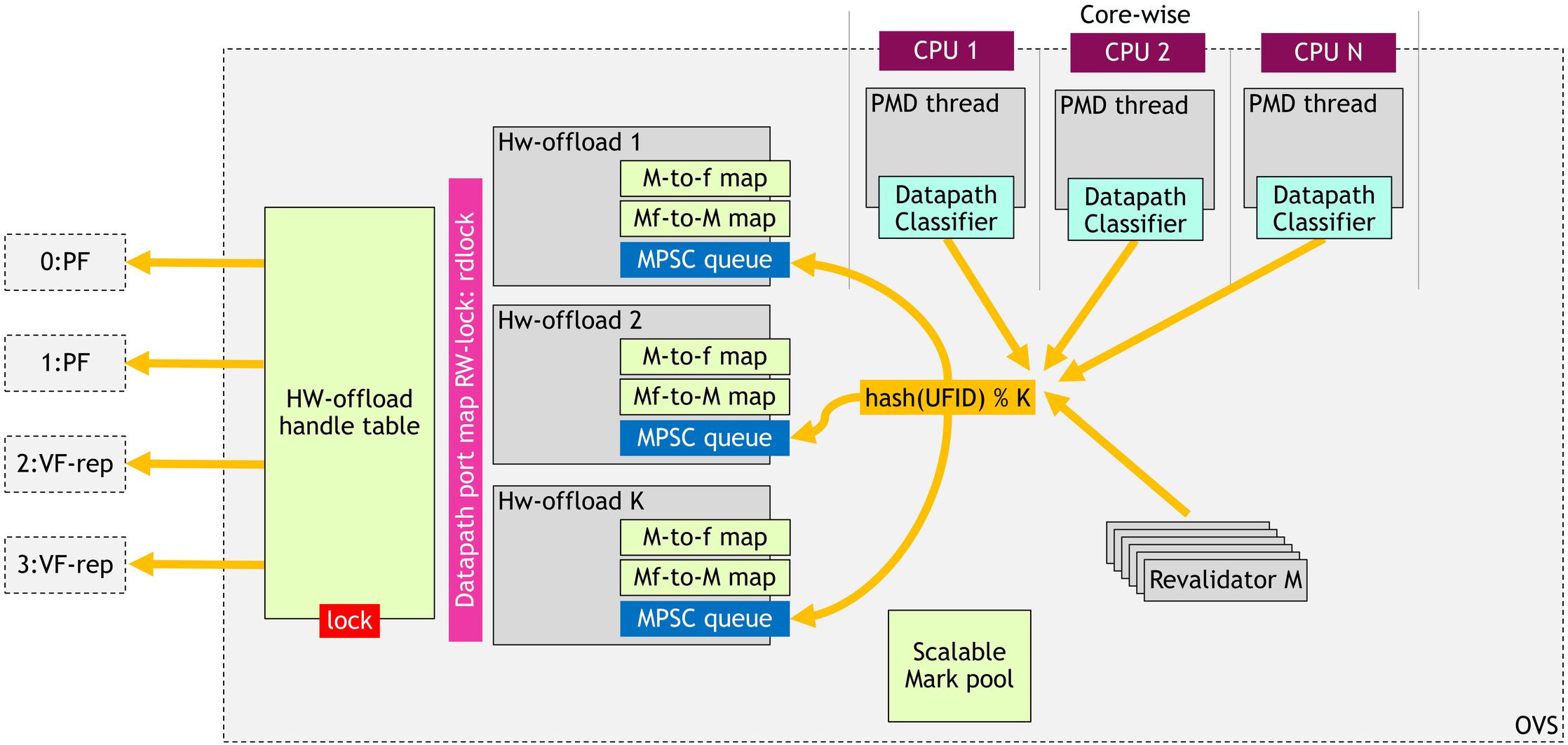
PARALLEL OFFLOAD INFRASTRUCTURE

Locking changes 3



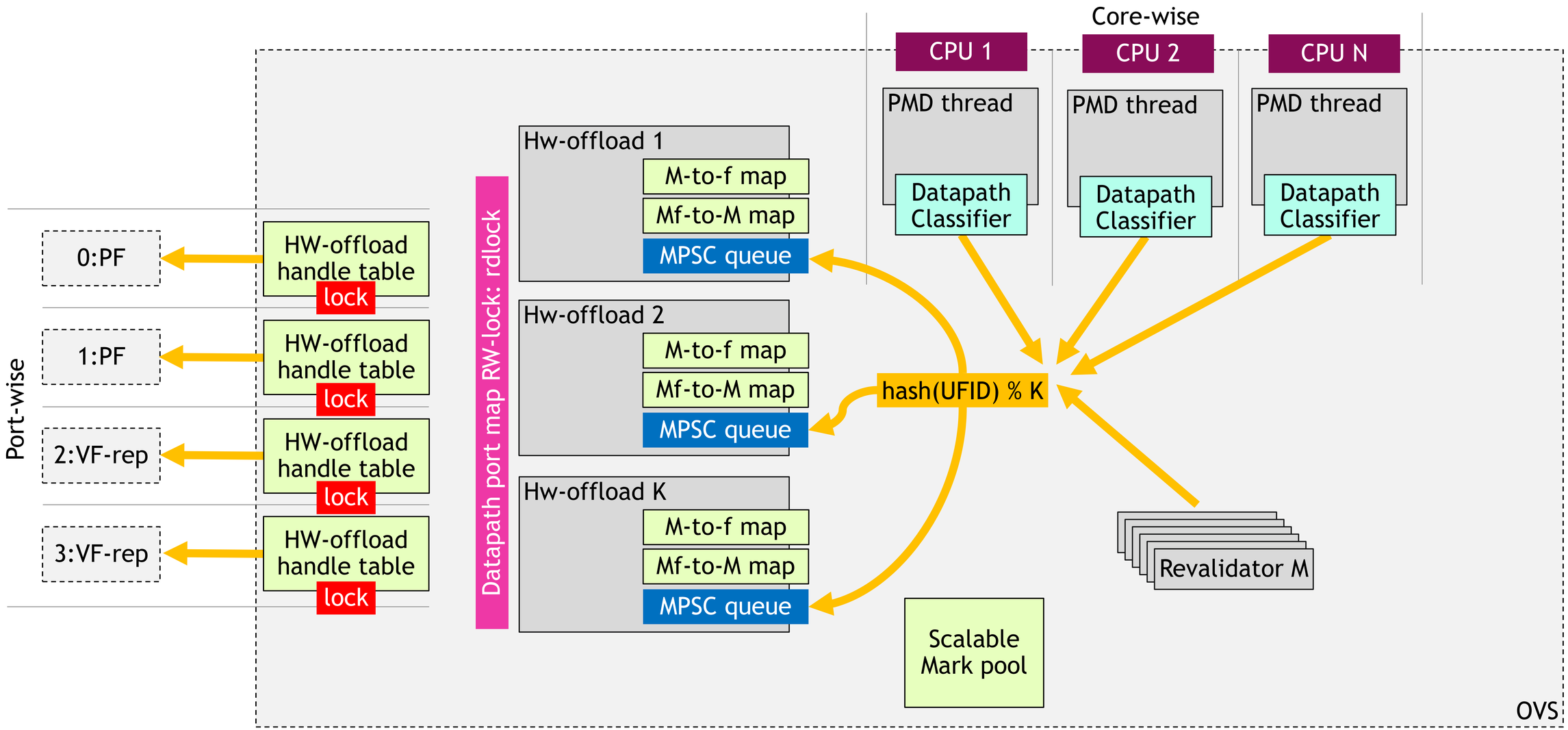
PARALLEL OFFLOAD ARCHITECTURE

Hw-offload thread pool



PARALLEL OFFLOAD ARCHITECTURE

Per-port offload maps



IMPLEMENTATION NOTES

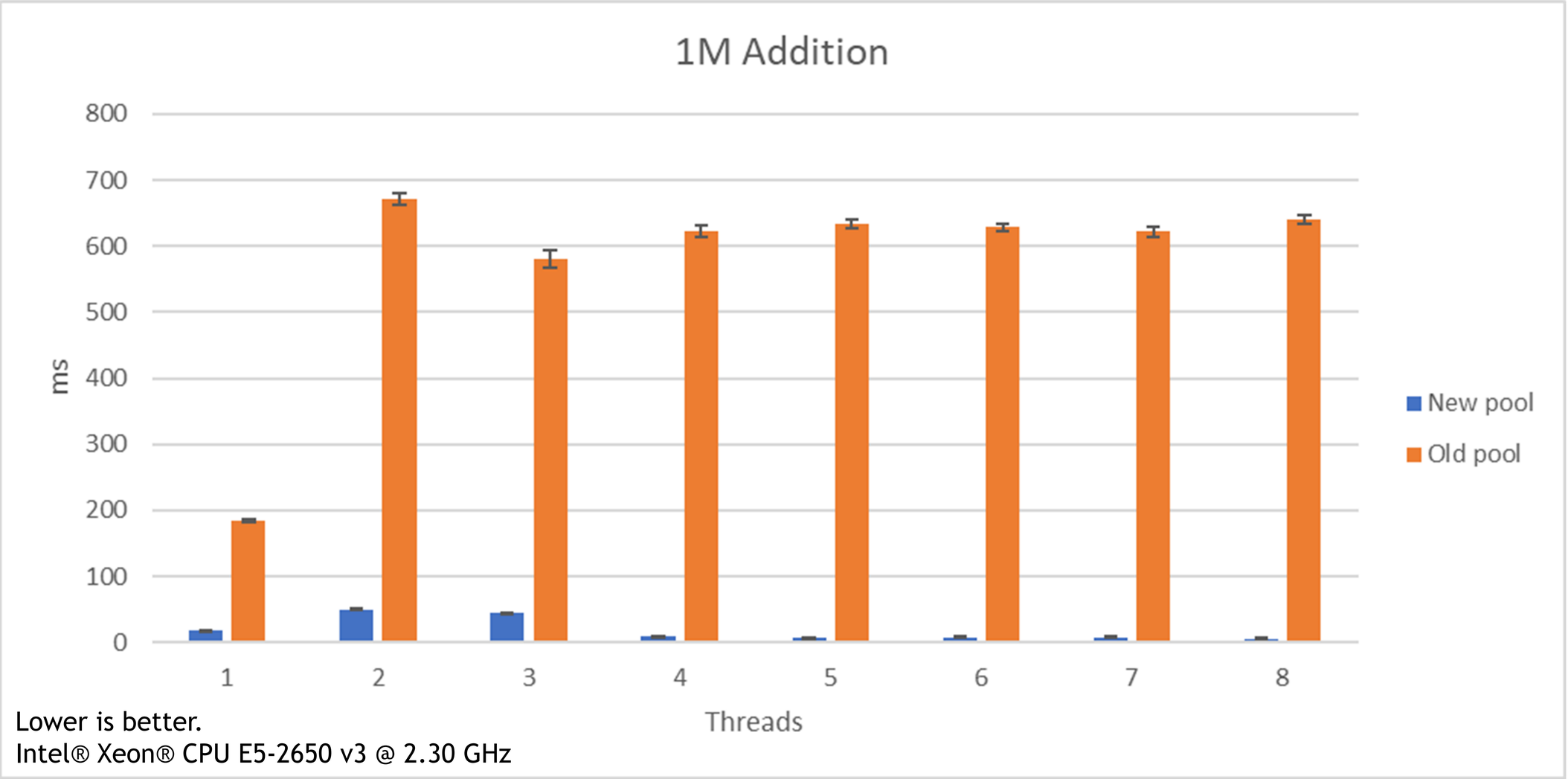
IMPLEMENTATION NOTES

- Fast Mark pool
  - Mark allocation uses an ID pool that has pathological behavior with non-sequential ID freeing.
  - This ID pool also scales poorly with additional threads.
  - It cannot be fixed without removing features used by other modules, that mark allocation does not require.

→ A new allocator is proposed: ‘id-fpool’. Functionalities are reduced to the essentials. It is faster and scales better.

IMPLEMENTATION NOTES

Faster mark pool: add



Lower is better.
Intel® Xeon® CPU E5-2650 v3 @ 2.30 GHz

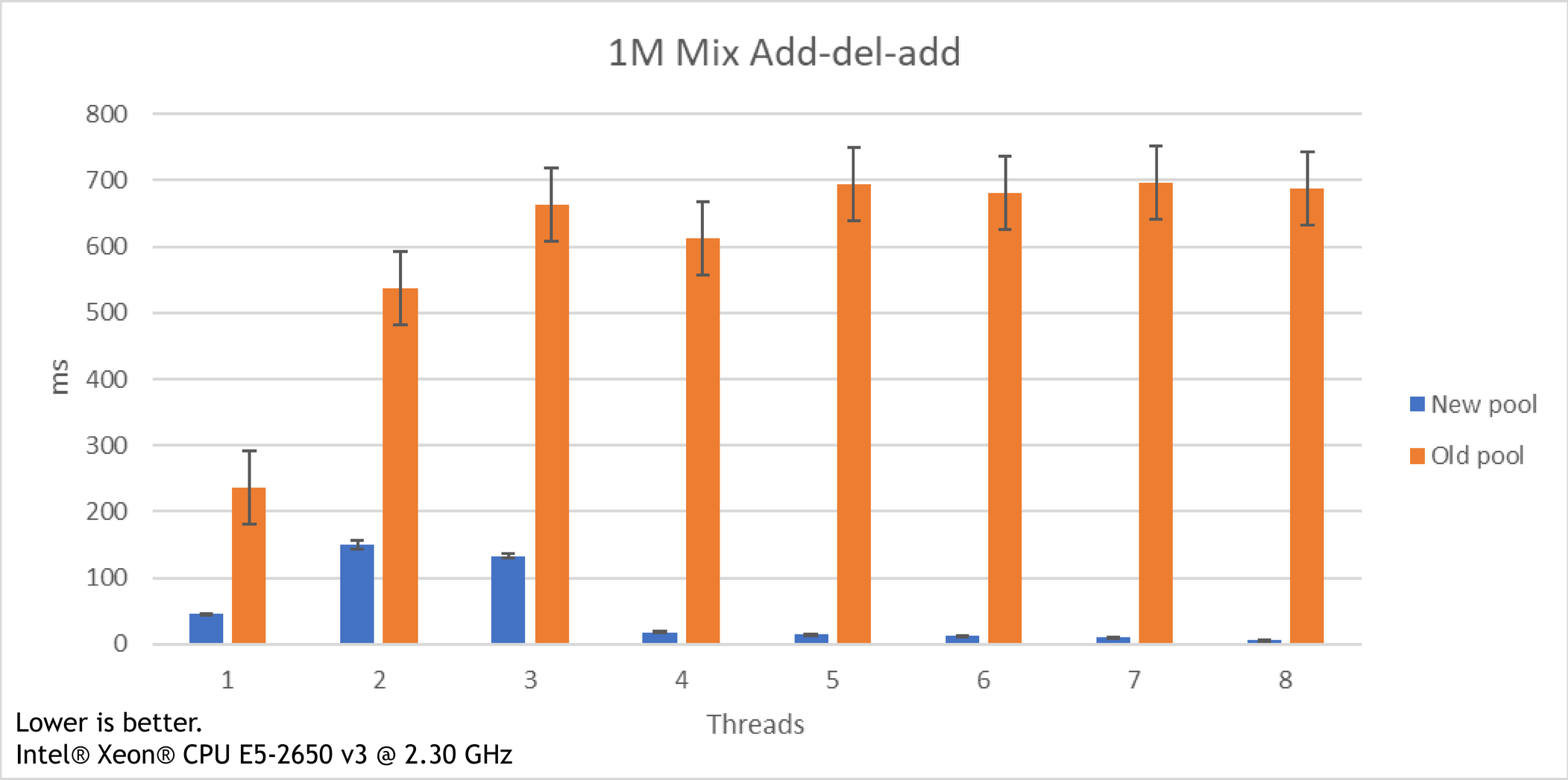
IMPLEMENTATION NOTES

Faster mark pool: del

Lower is better.
Intel® Xeon® CPU E5-2650 v3 @ 2.30 GHz

IMPLEMENTATION NOTES

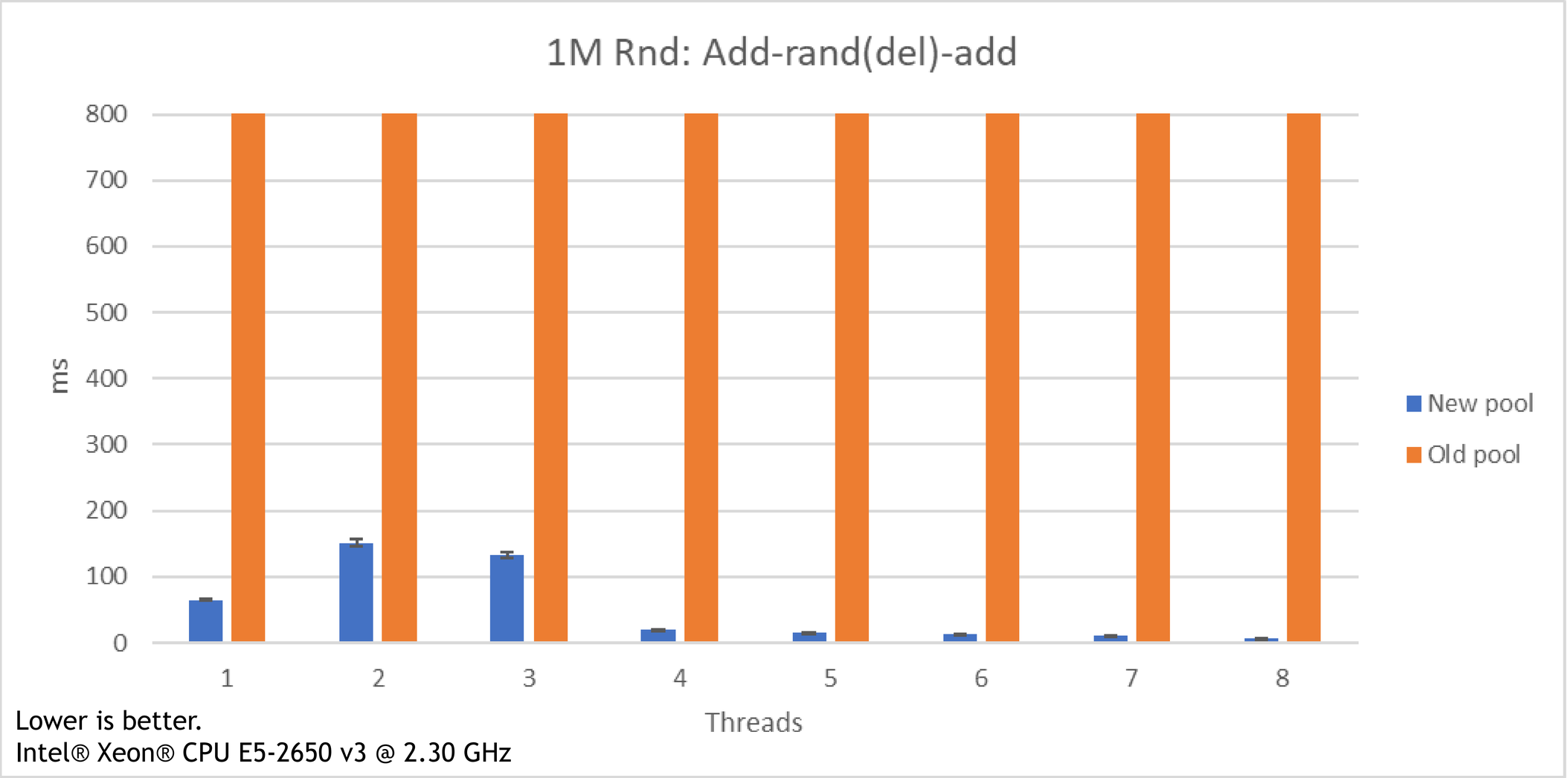
Faster mark pool: mix



Lower is better.
Intel® Xeon® CPU E5-2650 v3 @ 2.30 GHz

IMPLEMENTATION NOTES

Faster mark pool: rand



Lower is better.
Intel® Xeon® CPU E5-2650 v3 @ 2.30 GHz

IMPLEMENTATION NOTES

- Fast Mark pool
  - Mark allocation uses an ID pool that has pathological behavior with non-sequential ID freeing.
  - This ID pool also scales poorly with additional threads.
  - It cannot be fixed without removing features used by other modules, that mark allocation does not require.

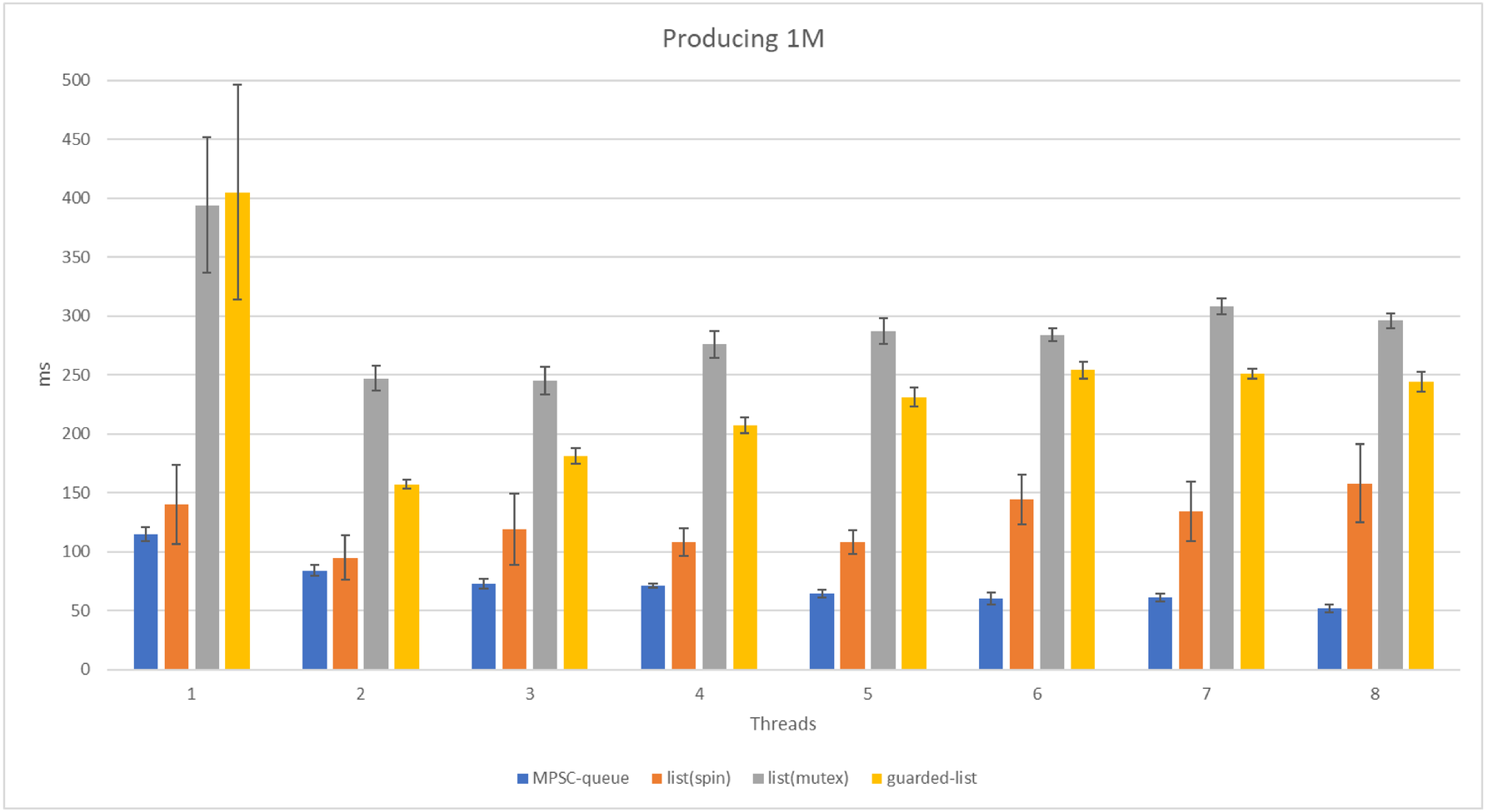
→ A new allocator is proposed: ‘id-fpool’. Functionalities are reduced to the essentials. It is faster and scales better.

- MPSC queue
  - Thread model is heterogeneous: Affined (PMD) and non-affined threads are all using the queue.
    Unfair lock (spinlock) is thus not usable.
    Fair lock (mutex) does not scale (+ worsen CPU coherency traffic).
  - Multi-Producer, Single-Consumer case is common.

→ A fast MPSC queue would be a useful addition to OvS.

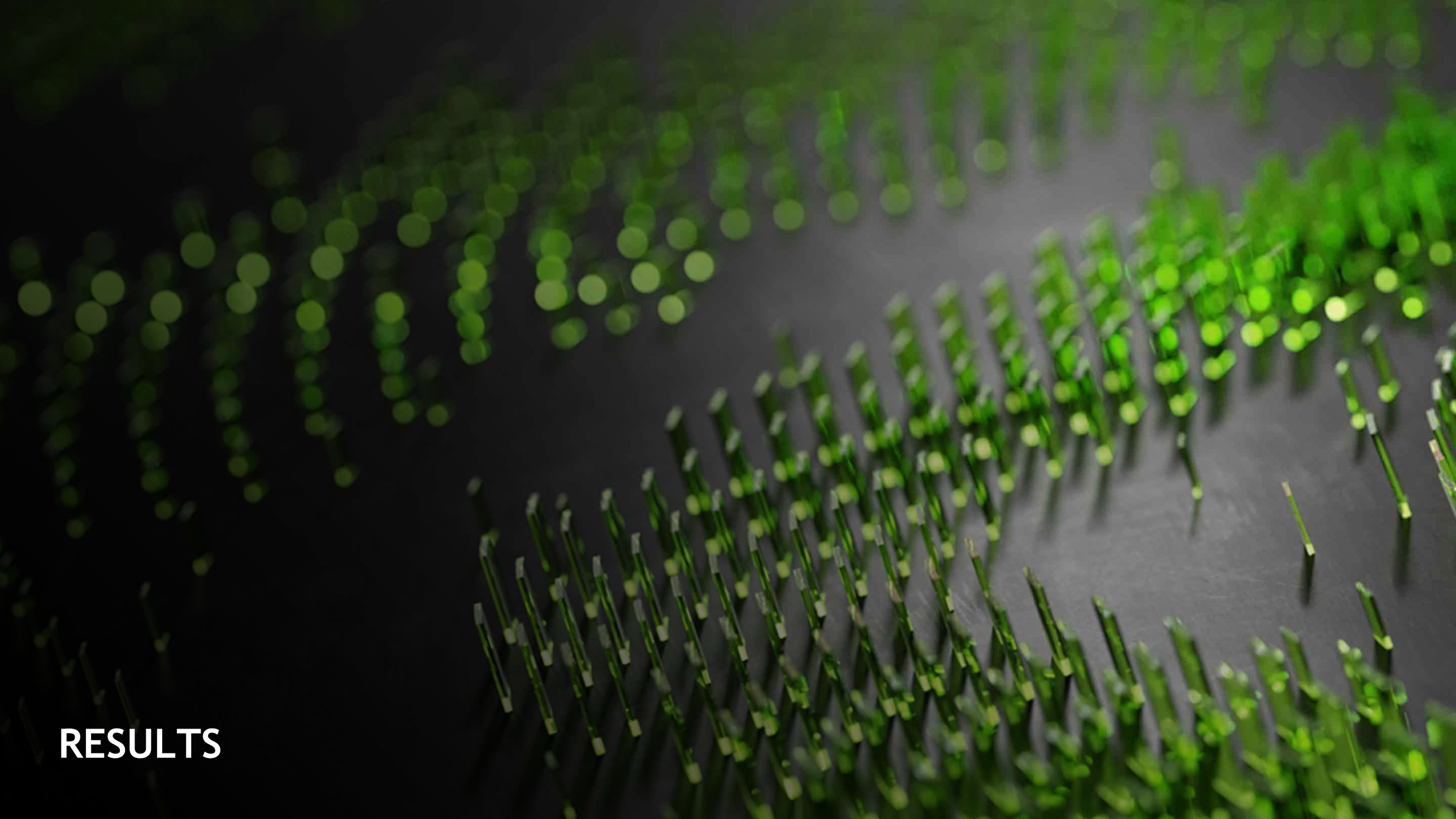
MPSC QUEUE

Producer side



Lower is better.

Intel® Xeon® CPU E5-2650 v3 @ 2.30 GHz

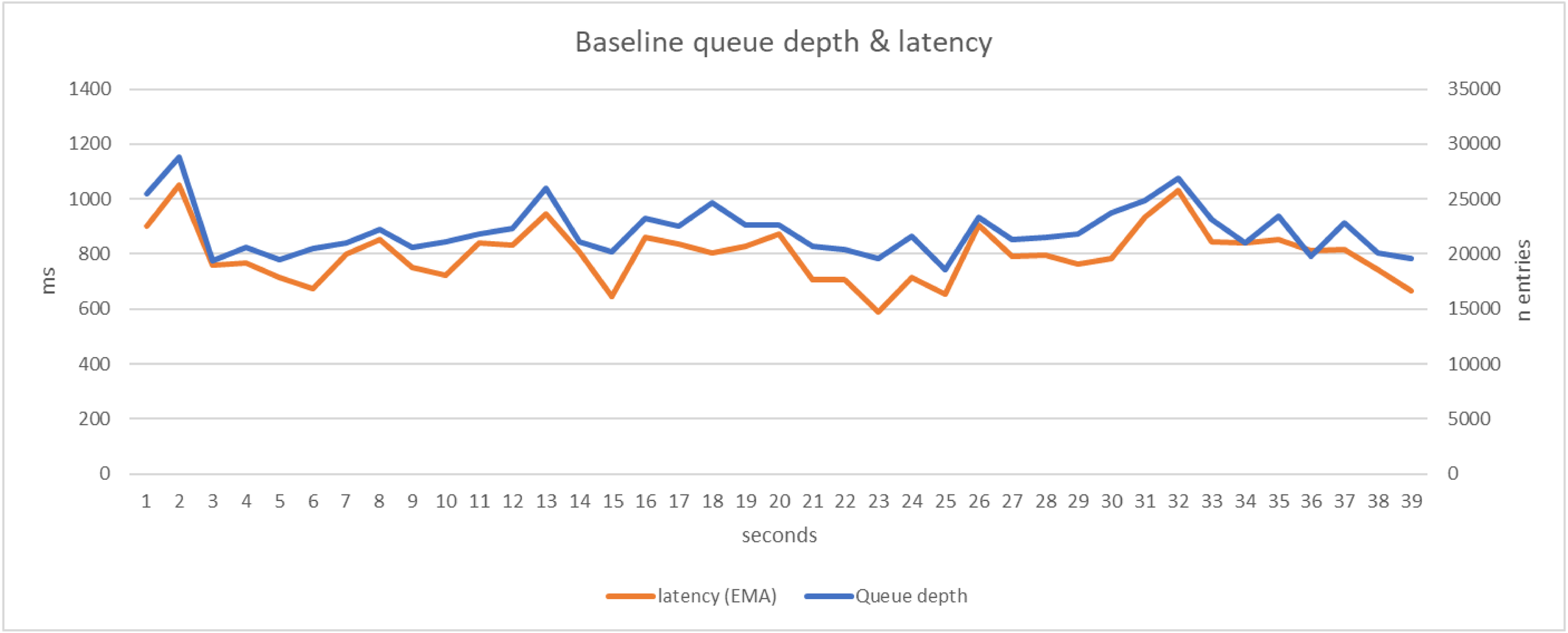

RESULTS

RESULTS

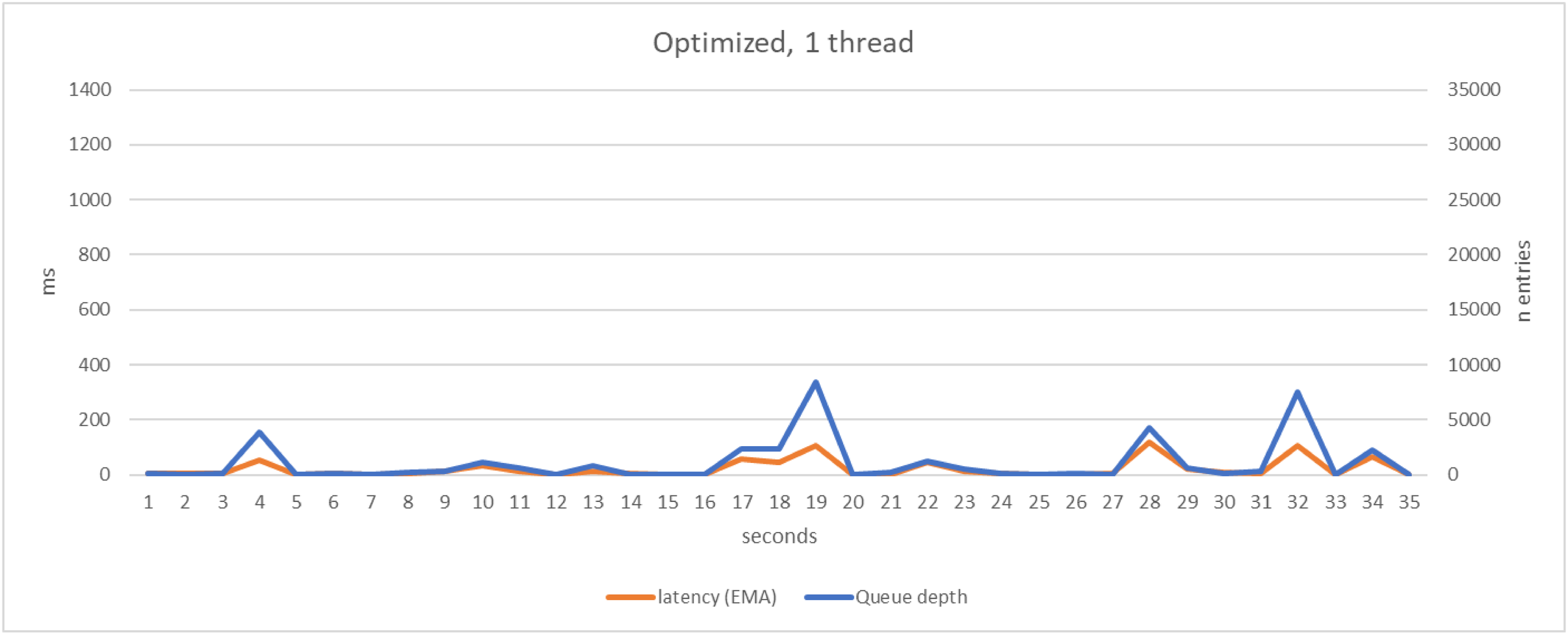
Setup

- Measures made on NVIDIA BlueField-2 DPU cores (Low power ARMv8.1 @ 500MHz).
- 4 PMD threads, 3 revalidators (default).
- Traffic made to trigger continuous updates.
- Latency is measured with an Exponential Moving Average, configured with a factor of 0.019802.
  - It gives the same ‘center of mass’ as a Simple Moving Average with a window of 100 entries.
  - Will respond quicker to recent changes, to show correlation with the offload queue depth.

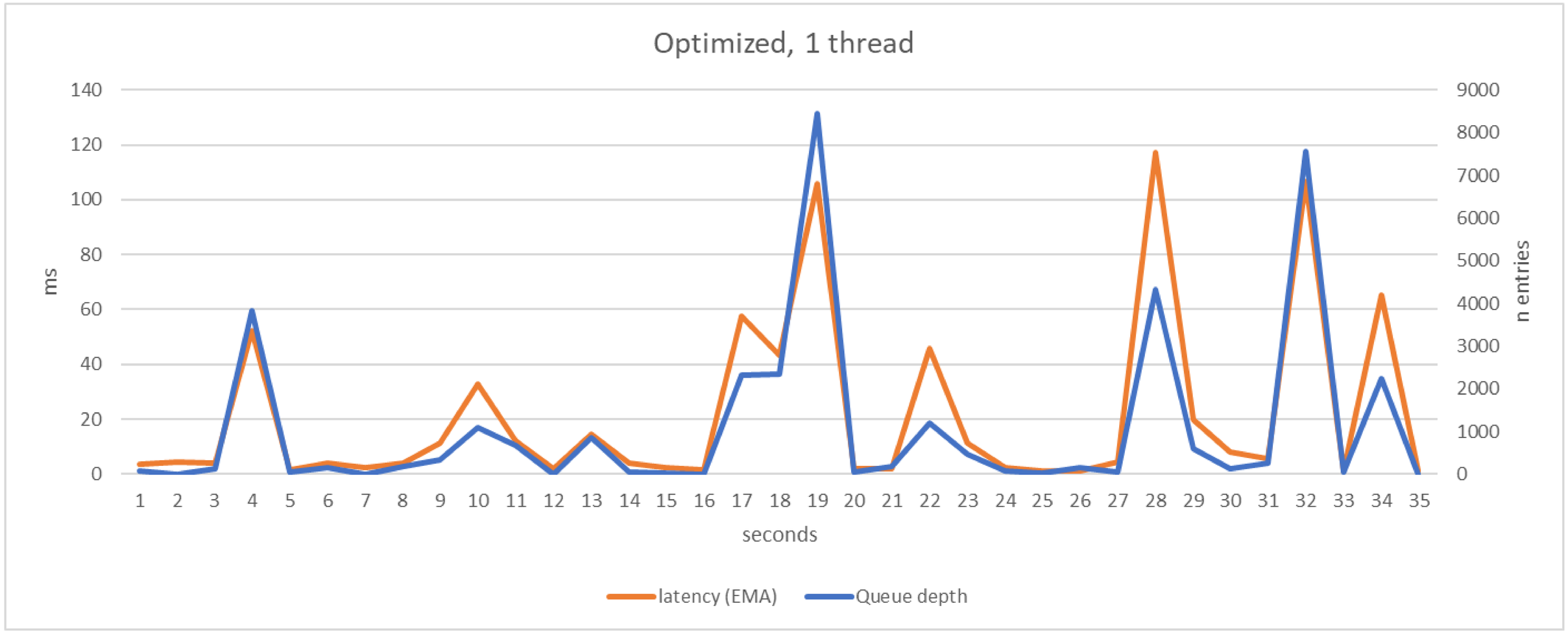
RESULTS



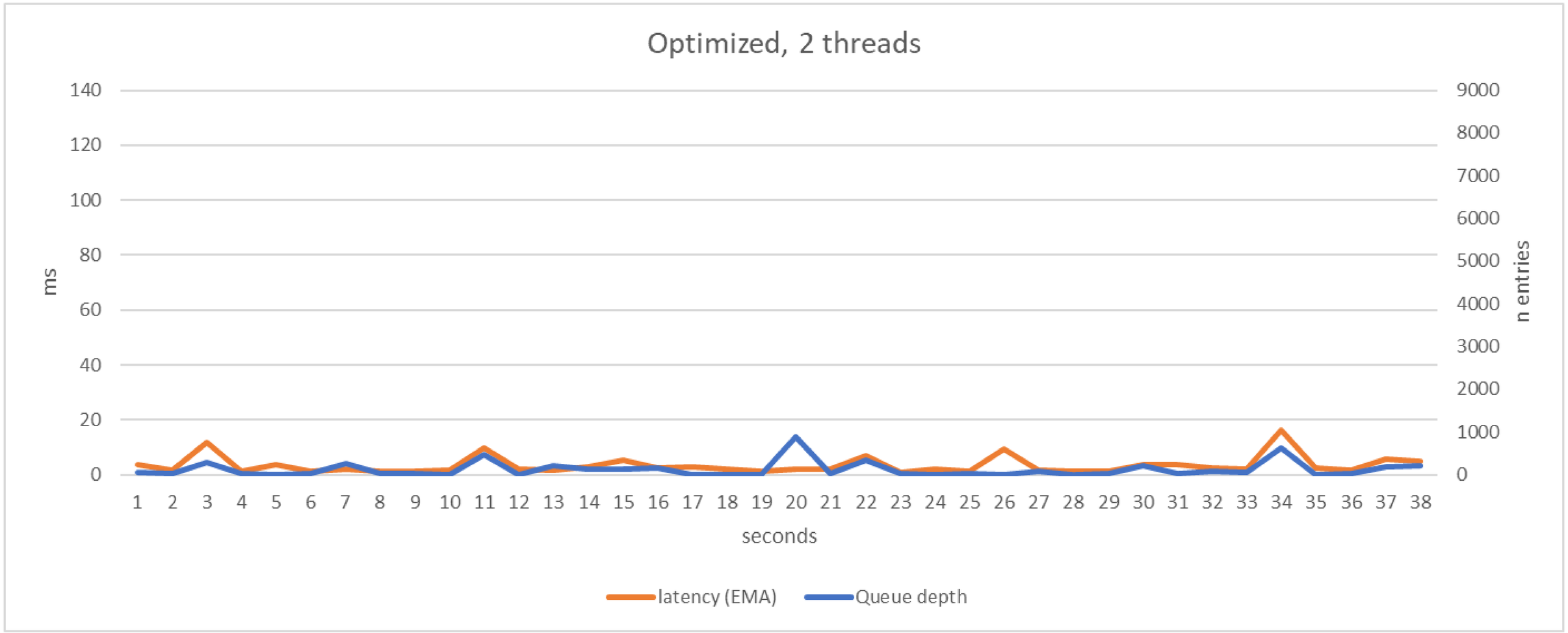
RESULTS



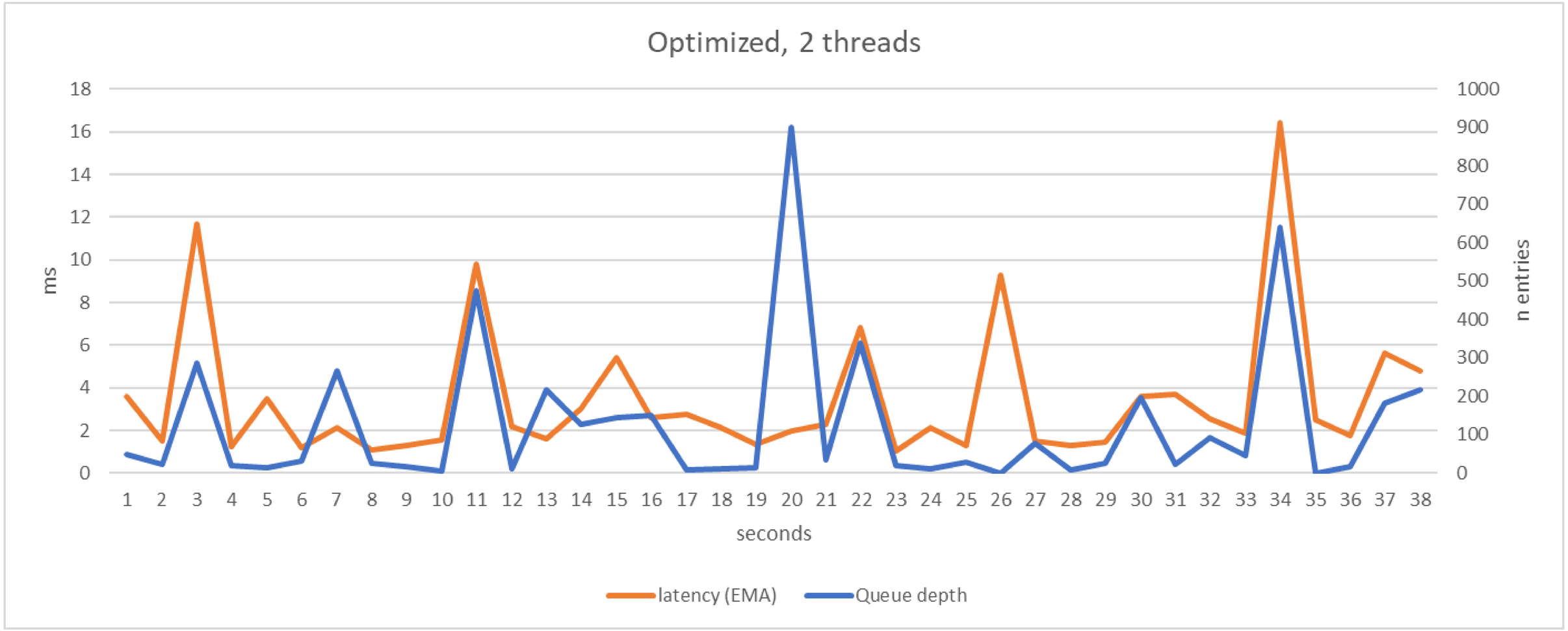
RESULTS



RESULTS



RESULTS


Optimized, 2 threads
ms
n entries
seconds
latency (EMA)
Queue depth

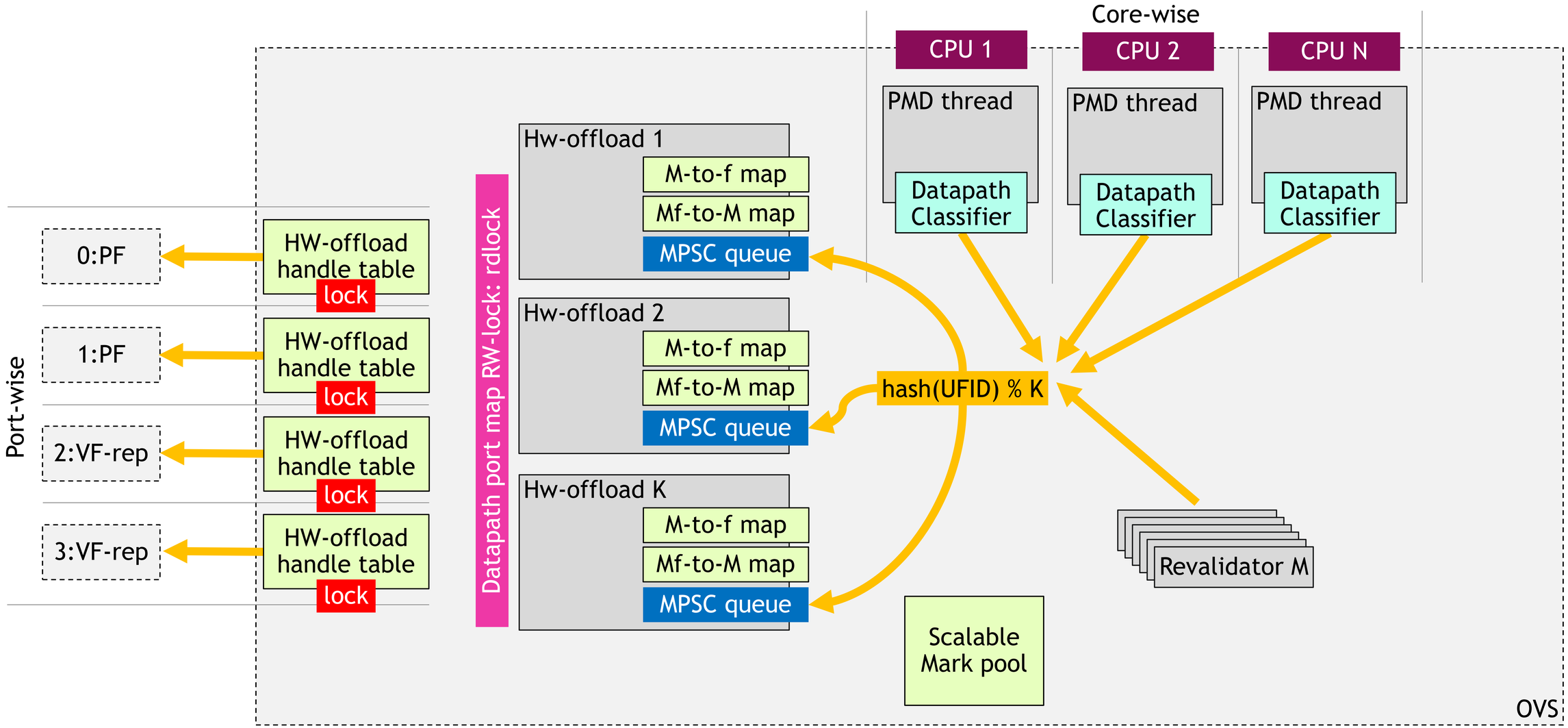
POSSIBLE IMPROVEMENTS

- Per-port offload table sharding.

  Divide each port offload table into K smaller tables, for K hw-offload threads.
  It would remove the last contention in the HW offload management.

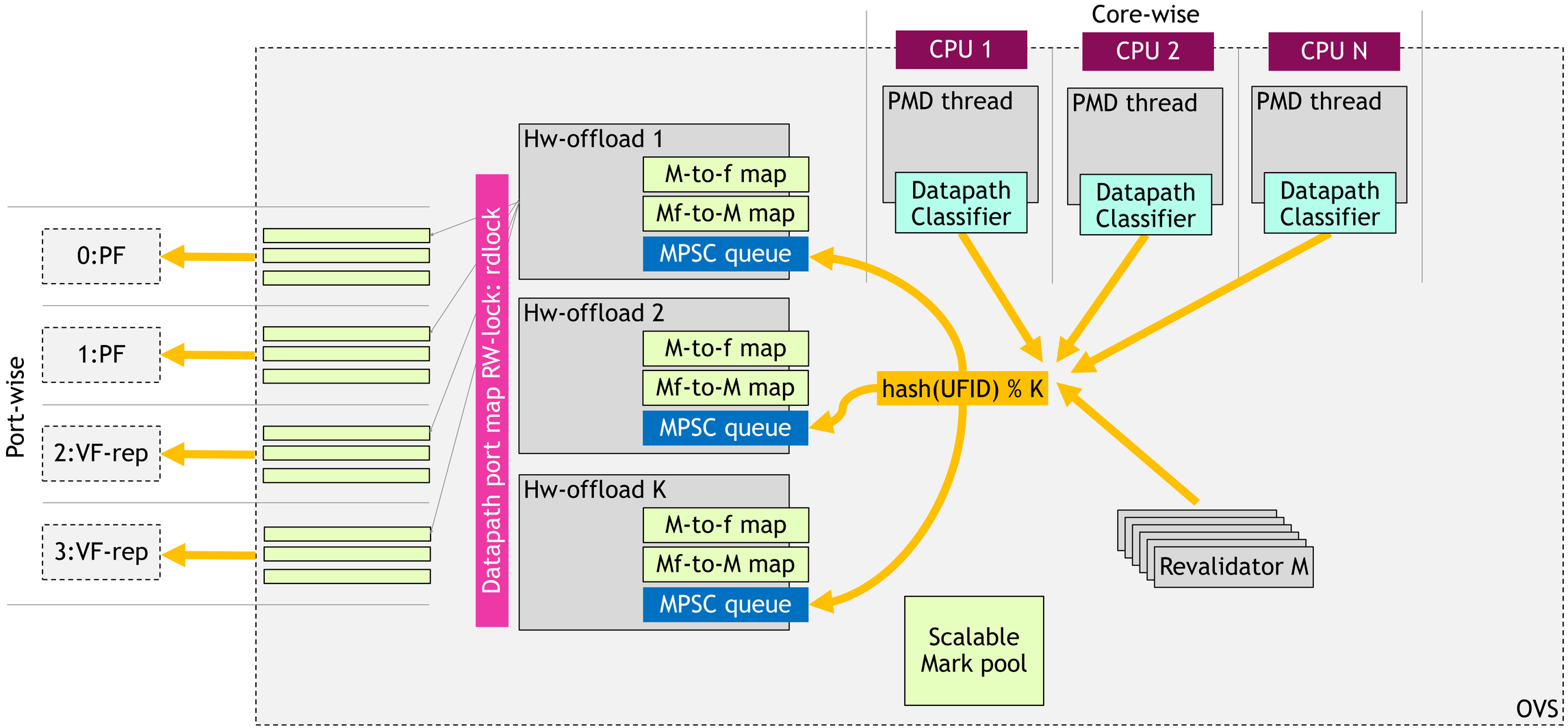
PARALLEL OFFLOAD ARCHITECTURE

Per-port offload maps



PARALLEL OFFLOAD ARCHITECTURE

Per-port offload maps



POSSIBLE IMPROVEMENTS

- Per-port offload table sharding.

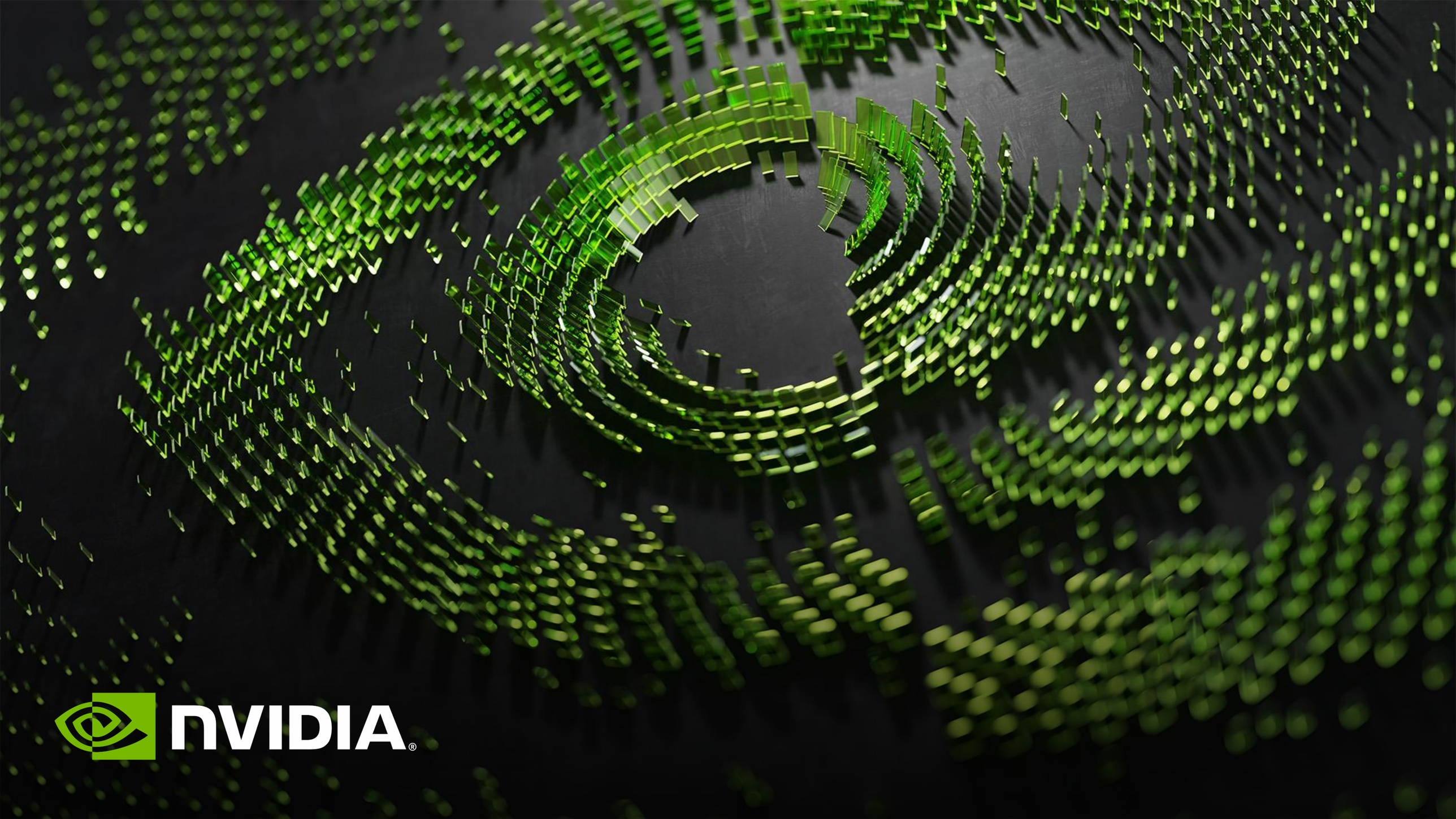
  Divide each port offload table into K smaller tables, for K hw-offload threads.
  It would remove the last contention in the HW offload management.

- Improve memory efficiency.

  Offloads data structure have been reorganized to allow parallel disjoint access.
  Beyond parallelism, memory access could be more cache-conscious:

  - spatially: reduce match footprint in offloads by avoiding a full description.
    Alternatively, offload match could be directly written by offload initiator (PMD / revalidator).

  - temporally: batch offload updates.


NVIDIA