DPDK
SUMMIT
NORTH AMERICA

# Zero-copy ring APIs with DPDK pipeline-mode applications

DHARMIK THAKKAR
ARM

JULY 12-13, 2021

# Agenda

DPDK SUMMIT NORTH AMERICA

- Implementation with traditional ring APIs
- Implementation with zero-copy ring APIs
- Test set-up
- Performance with scalar and vector PMDs – zero-copy vs traditional ring APIs
- Challenges with zero-copy ring APIs

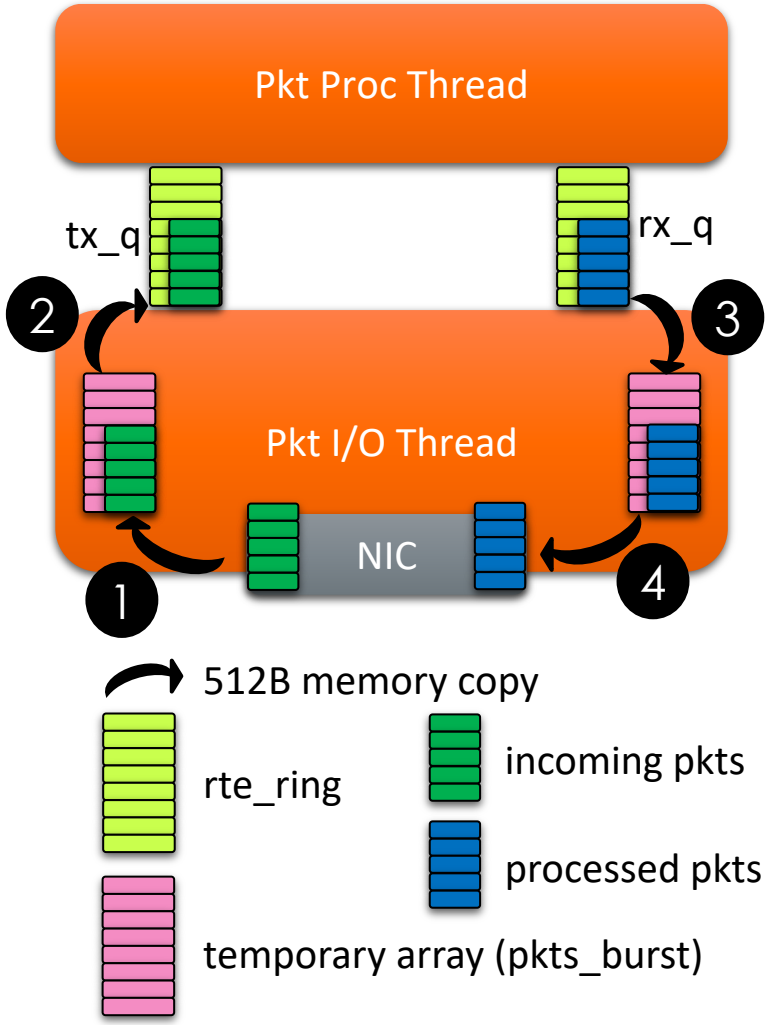
# Implementation with traditional ring APIs



- With traditional ring:

```
/* PKT I/O ENQUEUE */
nb_rx = rte_eth_rx_burst(portid, queueid,
                         pkts_burst, MAX_PKT_BURST);

ret = rte_ring_enqueue_burst(tx_q, (void**)pkts_burst,
                         nb_rx, NULL);

/* PKT I/O DEQUEUE */
ret = rte_ring_dequeue_burst(rx_q, (void**)pkts_burst,
                         MAX_PKT_BURST, NULL);

nb_rx = rte_eth_tx_burst(portid, queueid, pkts_burst, ret);
```

- mbufs are received in a temporary array (pkts_burst) during enqueue/dequeue operations

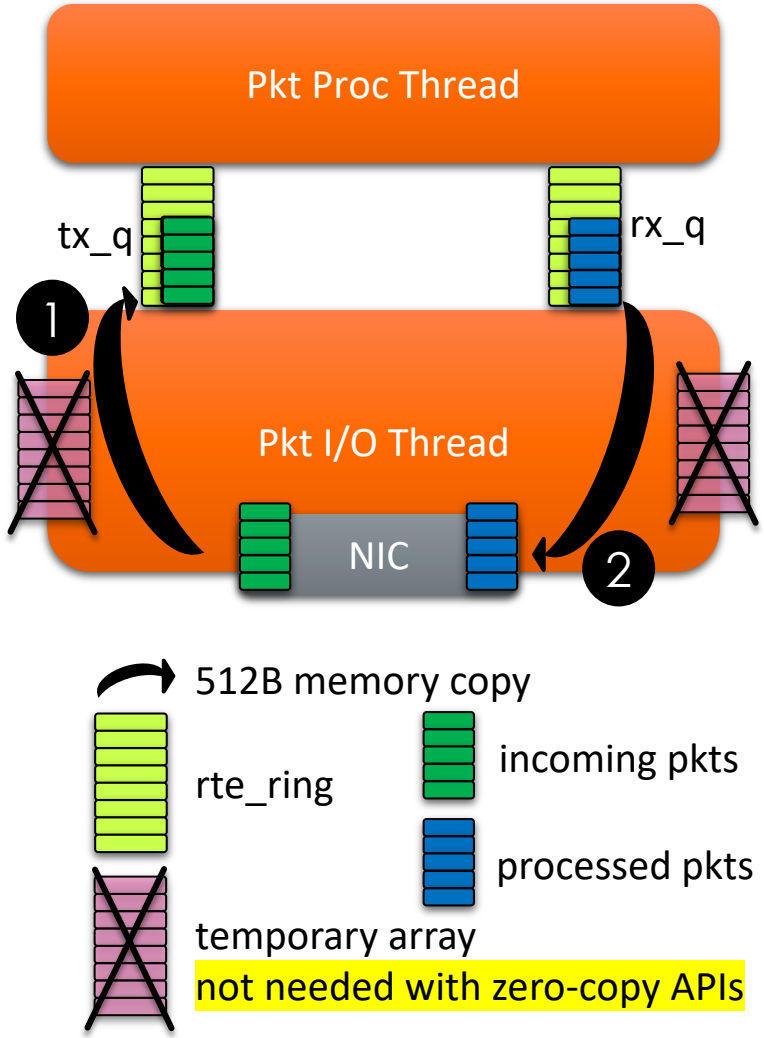


# Implementation with zero-copy ring APIs



- Zero-copy ring APIs help avoid intermediate copies by exposing the space on the ring directly to the application

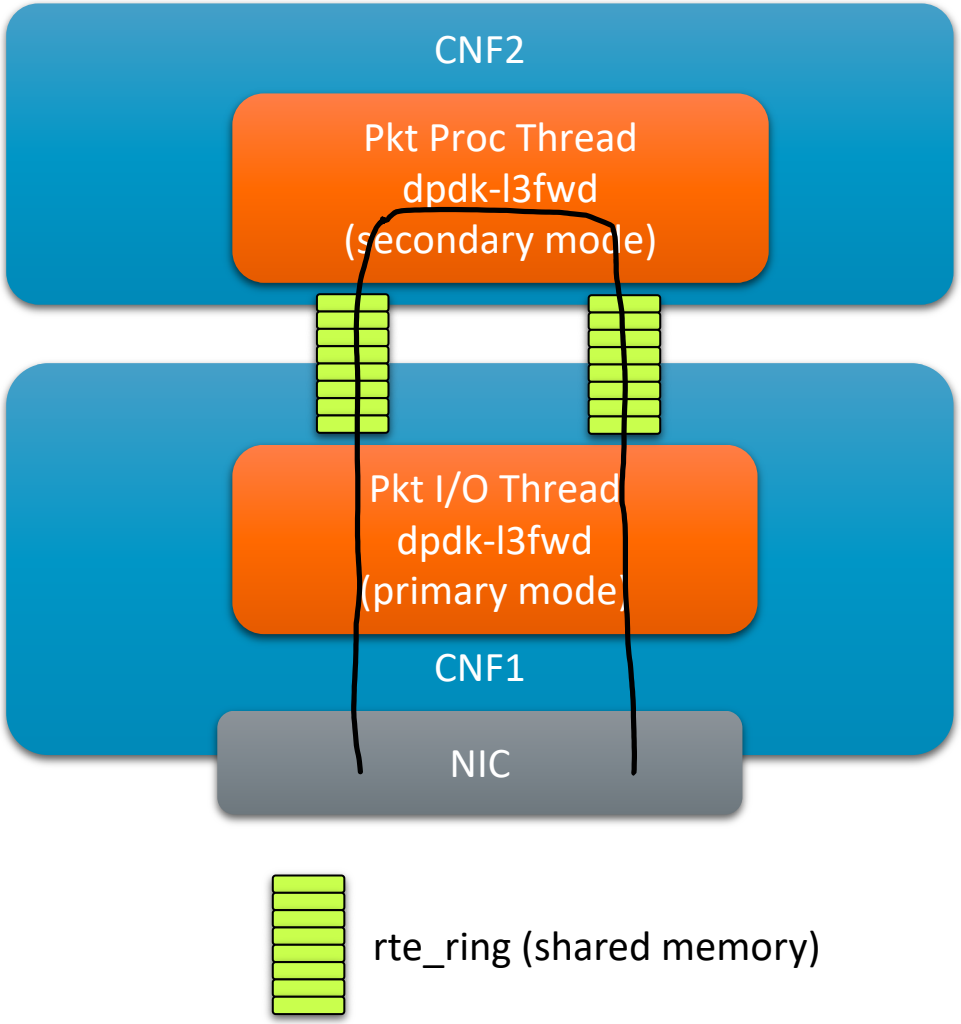
```
struct rte_ring_zc_data zcd;

/* PKT I/O ENQUEUE (ZERO_COPY_RING) */
n = rte_ring_enqueue_zc_burst_start(tx_q, MAX_PKT_BURST,
                                    &zcd, NULL);

nb_rx = rte_eth_rx_burst(portid, queueid,
                         zcd.ptr1, zcd.n1);

rte_ring_enqueue_zc_finish(tx_q, nb_rx);

/* PKT I/O DEQUEUE (ZERO-COPY RING) */
ret = rte_ring_dequeue_zc_burst_start(rx_q, MAX_PKT_BURST,
                                      &zcd, NULL);

nb_tx = rte_eth_tx_burst(portid, queueid,
                         zcd.ptr1, zcd.n1);

rte_ring_dequeue_zc_finish(rx_q, nb_tx);
```



# Test set-up



- Compare performance of zero-copy and traditional ring APIs
- L3FWD pipeline with pkt I/O and pkt proc stages
- CNF1 - PKT I/O Thread – RX/TX packets from/to the NIC
- CNF2 – PKT Processing Thread – RX/TX packets from/to CNF1 via ring APIs (shared memory)



# Performance with scalar PMDs



- Throughput (MPPS):

| | Throughput |
|---|---|
| **Traditional ring** | 23.01 |
| **Zero-copy ring** | 24.29 |

- 5.5% improvement in throughput with zero-copy ring APIs

- Kernel PMU Events (Per packet):

| PMU Events | Traditional Ring | Zero-copy Ring | % improvement |
|---|---|---|---|
| stalled-cycles-backend * | 84.23 | 74.46 | -11.60 |
| stalled-cycles-frontend * | 1.13 | 0.86 | -23.75 |
| l1d_cache_refill (misses) * | 8.99 | 9.15 | 1.80 |
| l2d_cache_refill (misses) * | 4.26 | 3.99 | -6.32 |
| l3d_cache_refill (misses) * | 3.80 | 3.66 | -3.70 |
| l2d_cache_wb (evictions) * | 4.31 | 4.23 | -1.75 |
| l1d_cache_wb (evictions) * | 7.44 | 7.38 | -0.73 |
| mc_reqs (DRAM access) * | 6.13 | 6.42 | 4.71 |
| IPC | 4.36 | 4.37 | 0.41 |

* smaller is better

# Performance with vector PMDs



- Throughput (MPPS):

| | Throughput |
|---|---|
| **Traditional ring** | 25.65 |
| **Zero-copy ring** | 28.16 |

- 9.8% improvement in throughput with zero-copy ring APIs

- Kernel PMU Events (Per packet):

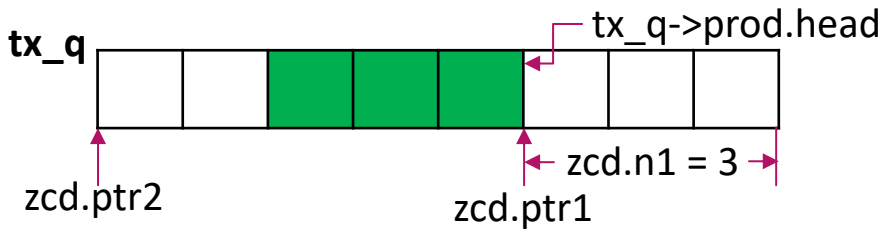
| PMU Events | Traditional Ring | Zero-copy Ring | % improvement |
|---|---|---|---|
| stalled-cycles-backend * | 92.72 | 89.58 | -3.38 |
| stalled-cycles-frontend * | 5.03 | 3.36 | -33.26 |
| l1d_cache_refill (misses) * | 9.30 | 9.08 | -2.37 |
| l2d_cache_refill (misses) * | 5.06 | 4.88 | -3.53 |
| l3d_cache_refill (misses) * | 4.04 | 3.46 | -14.31 |
| l2d_cache_wb (evictions) * | 3.50 | 3.32 | -5.06 |
| l1d_cache_wb (evictions) * | 7.62 | 6.93 | -8.96 |
| mc_reqs (DRAM access) * | 5.78 | 5.59 | -3.21 |
| IPC | 3.19 | 2.93 | -8.16 |

* smaller is better

# Challenges with zero-copy ring APIs



- Wrap-around case with vector PMDs:




```
/* PKT I/O Rx loop (zero-copy mode) */
n = rte_ring_enqueue_zc_burst_start(tx_q, MAX_PKT_BURST,
                                    &zcd, NULL);
```

- Vectorized Rx PMDs cannot receive less than

  xxx_DESCS_PER_LOOP, which is equal to 4 for mlx5, i40e, etc.

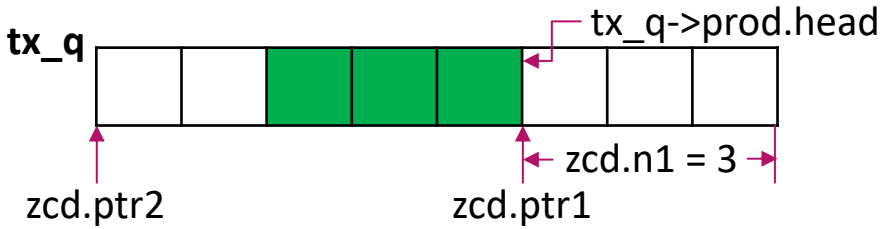
  Thus, any attempt to receive a burst of pkts in such a case fails

```
/* API returns 0 RX pkts*/
nb_rx = rte_eth_rx_burst(portid, queueid,
                         zcd.ptr1, zcd.n1);
```

# Challenges with zero-copy ring APIs



- Solution: Hybrid (copy. + zero-copy)
  - Pkts can be first received in a temporary array and then copied onto the ring

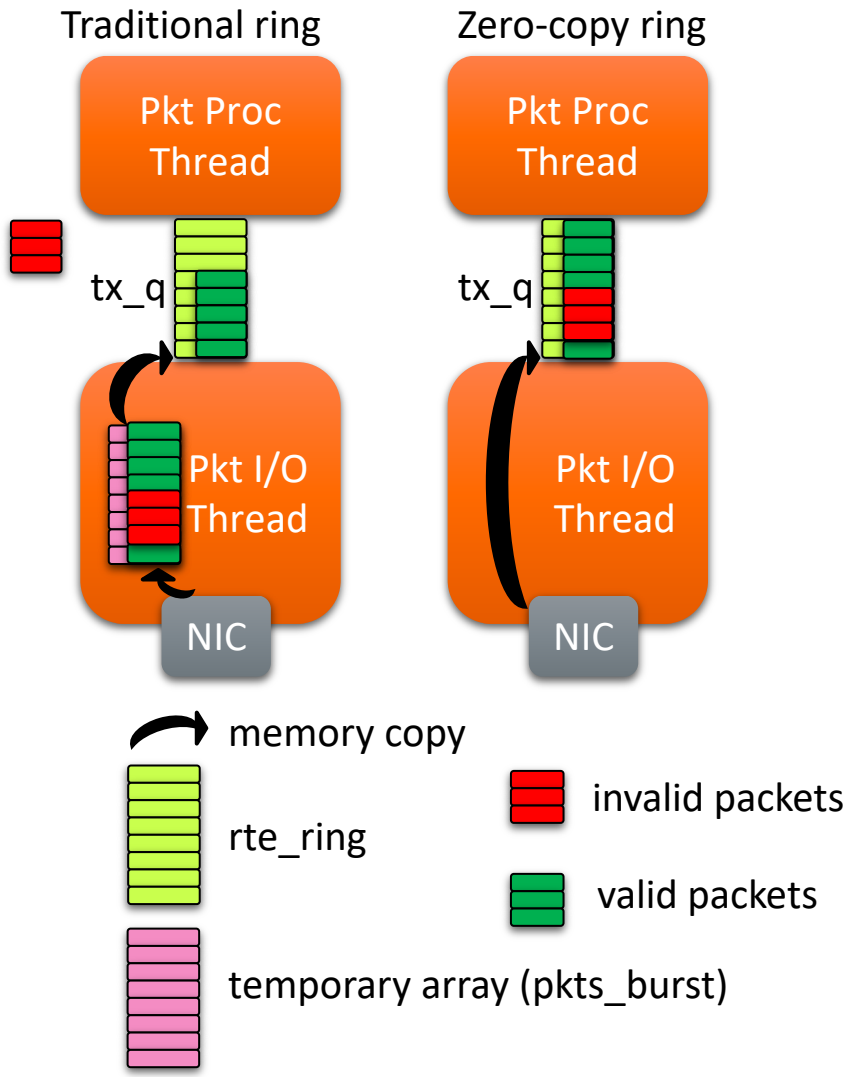



```
/* PKT I/O Rx loop (copy + zero-copy) */
n = rte_ring_enqueue_zc_burst_start(tx_q, MAX_PKT_BURST, &zcd, NULL);

if (zcd.ptr2 != NULL) { /* Wrap around */
    nb_rx = rte_eth_rx_burst(portid, queueid, pkts_burst,
                             MAX_PKT_BURST);
    rte_memcpy(zcd.ptr1, pkts_burst, zcd.n1 * 8);
    rte_memcpy(zcd.ptr2, pkts_burst + zcd.n1, (nb_rx - zcd.n1) * 8);

} else { /* Normal case */
    nb_rx = rte_eth_rx_burst(portid, queueid, zcd.ptr1, zcd.n1);
}
```

# Challenges with zero-copy ring APIs



- Requires careful code refactoring
  - Invalid packets need a complex freeing mechanism
- APIs are available only for two sync modes:
  - Single Producer/Single Consumer (RTE_RING_SYNC_ST)
  - Serialized Producer/Serialized Consumer (RTE_RING_SYNC_MT_HTS)



Q & A

DPDK

SUMMIT

NORTH AMERICA

Thank you!