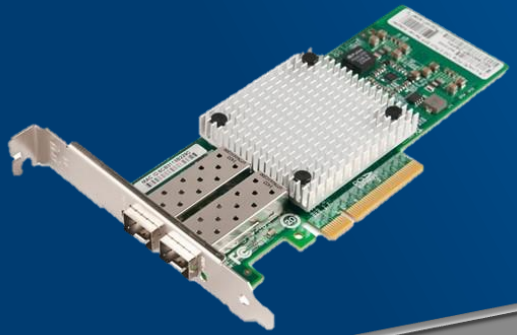
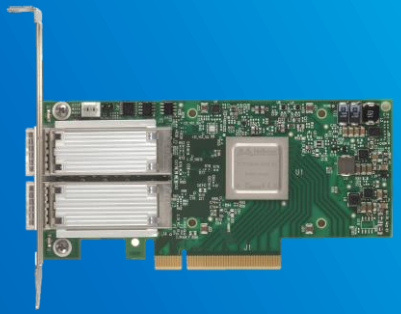
NETWORK PERFORMANCE WORKSHOP

Alexander Duyck
Open Source Technologist
Intel Corporation
Alexander.Duyck@gmail.com





NETDEV 2.1

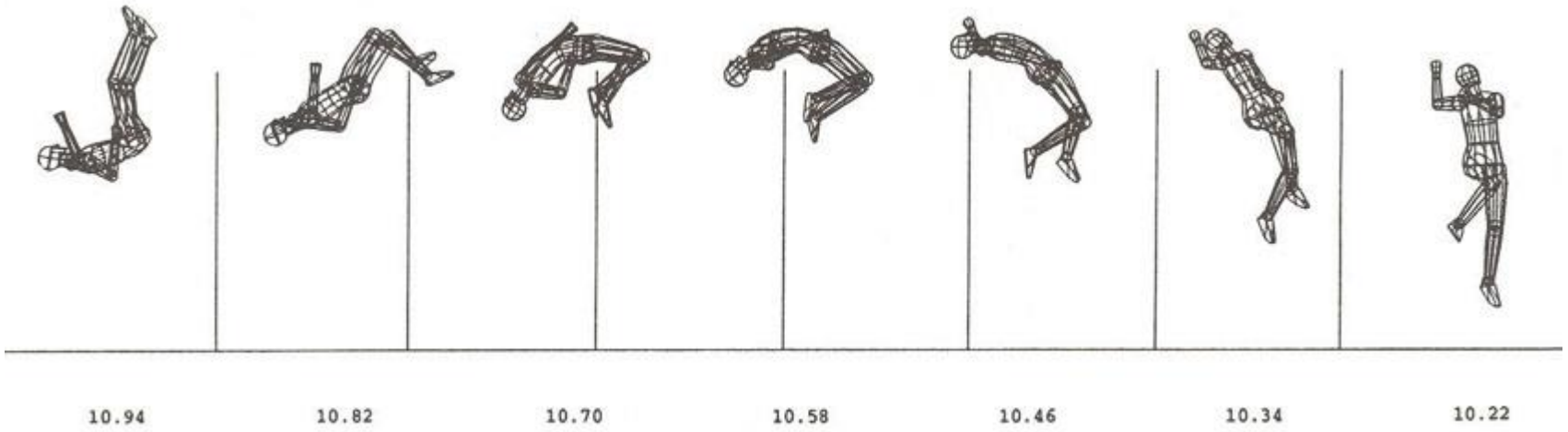
PERFORMANCE DEATHMATCH

Agenda

- Alexander Duyck
  - Implementing Page Based Receive Paths with Page Reuse
  - DMA API Changes to Enable Better Use of build_skb and XDP
- Jesper Dangaard Brouer
  - Kernel Memory Optimizations
- John Fastabend
  - Zero-copy Using AF_PACKET to Accelerate User-Space Networking
- Amir Ancel, Saeed Mahameed, Tariq Toukan
  - Rx Streaming
  - Tx Bulking
  - Multi packet Tx Descriptor

Coopetition

- Fosbury Flop



IMPLEMENTING PAGE BASED RECEIVE PATHS WITH PAGE REUSE

Basics of Page Based Receive

1. Alloc Page
2. Map Page
3. Assign Page to Device
4. Unmap Page
5. Assign Page to skb using build_skb or skb_add_rx_frag
6. Return to step 1

Basics of Page Based Receive with Page Reuse

1. Alloc Page
2. Map Page
3. Assign Page to Device

4.A. Sync Half of Page for CPU

5.A. Assign Page to skb using skb_add_rx_frag - (read only)

6.A. Increment Page Count Using get_page() or page_ref_add()

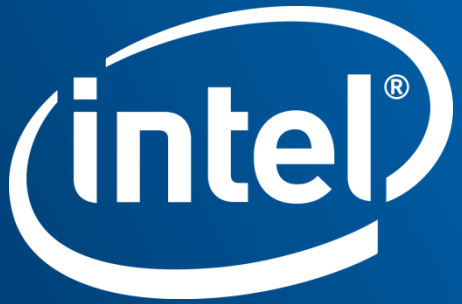
7. Sync Other Half of Page for Device, or use __page_frag_cache_drain() to free
8. Return to Step 3 or 1 depending on page state

Drop the Read Only Requirement with DMA_ATTR_SKIP_CPU_SYNC

- Pages were read-only as dma_unmap_page() would invalidate data
- Adding DMA_ATTR_SKIP_CPU_SYNC as DMA attribute to map/unmap prevents this
- Drivers required to use dma_sync_for_cpu/device
- Code already in igb, ixgbe, and soon i40e/i40evf

Use Memory Barriers Responsibly

- Many drivers still using wmb() or rmb() to guarantee ordering
  - Causes pipeline stalls
  - Only needed to guaranteed ordering between MMIO and coherent memory
- The dma_wmb() and dma_rmb() barriers can provide mem vs mem ordering
  - On x86 they convert to barrier() and compile out
  - On other architectures they are still less expensive than wmb()/rmb()


intel®