TRANSFORMING COMMUNICATIONS

Network Product Group

The Data Plane Development Kit (DPDK) – What it is and where it's going

John Ronciak, John Fastabend, Danny Zhou, Mark Chen, Cunming Liang

Intel® Data Plane Development Kit

What is the Intel® **DPDK**?

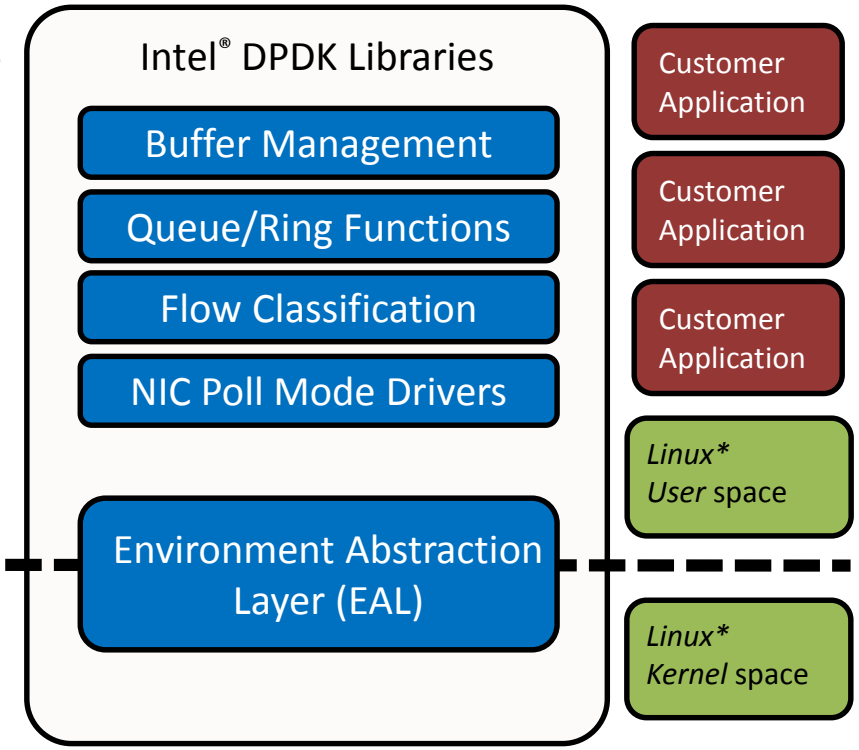
- A set of optimized software libraries and drivers that can be used to accelerate packet processing on Intel® architecture
  - *Packets are delivered into user space directly*

BSD Licensed with source available

- *Offered as a free, unsupported standalone solution by Intel or as part of commercial solutions from leading ecopartners*

Intel® DPDK Fundamentals

- Implements a run to completion model or pipeline model
- No scheduler - all devices accessed by polling
- Supports 32-bit and 64-bit with/without NUMA
- Scales from Intel® Atom™ to Intel® Xeon® processors
- Number of Cores and Processors not limited
- Optimal packet allocation across DRAM channels
- Use of 2M & 1G hugepages and cache align structures

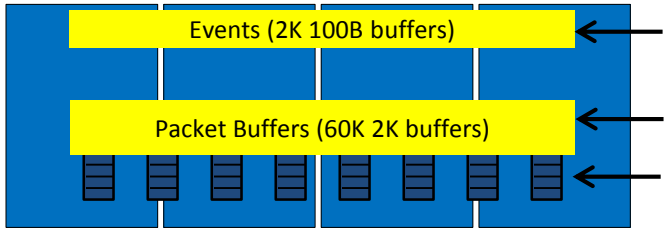


The Intel® DPDK embeds optimizations for the Intel® architecture platform, providing breakthrough packet processing performance

*Other names and brands may be claimed as the property of others.

Intel® DPDK model

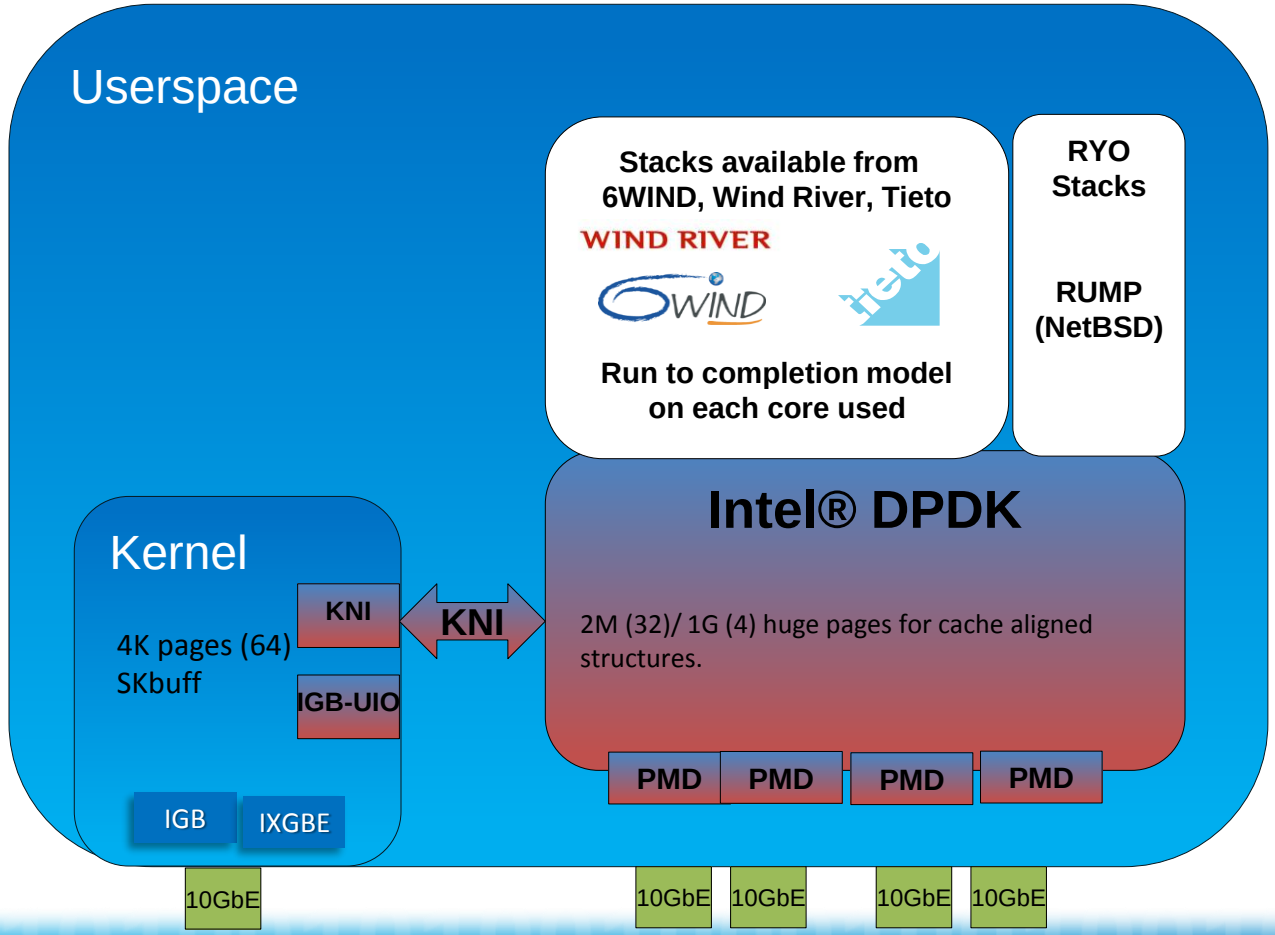
Intel® DPDK allocates packet memory equally across 2, 3, 4 channels.
Aligned to have equal load over channels

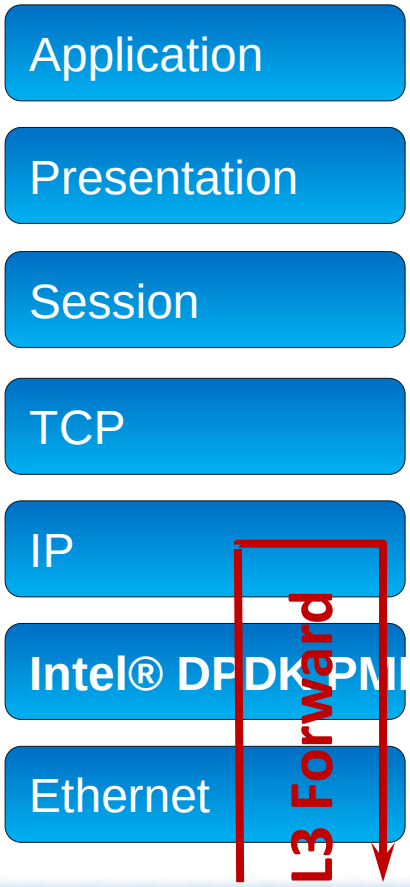


Mempools (Ring) for Events, Msgs, etc.

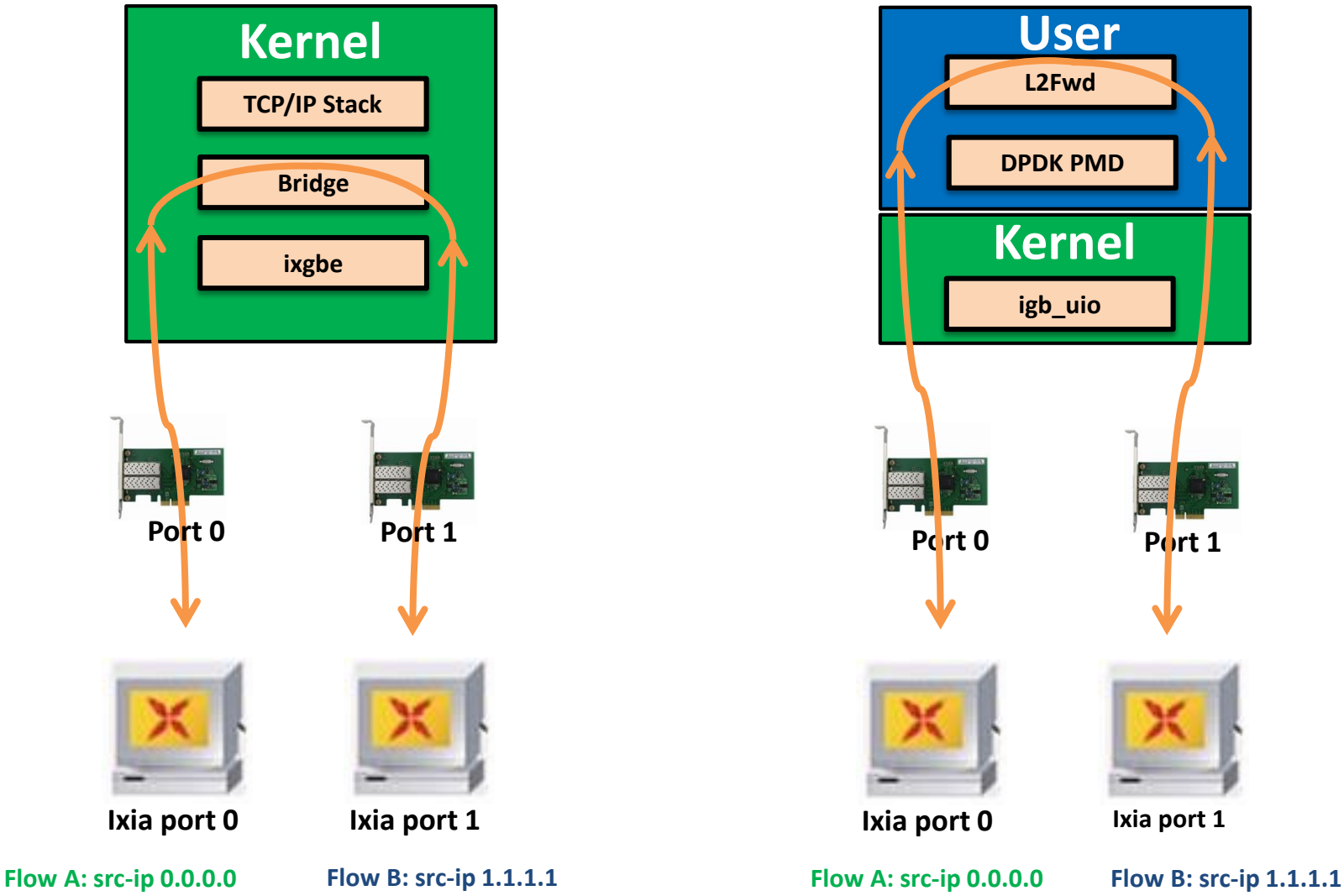
Mempool (Ring) for cached buffers

Per core lists, unique per lcore. Allows packet movement without locks



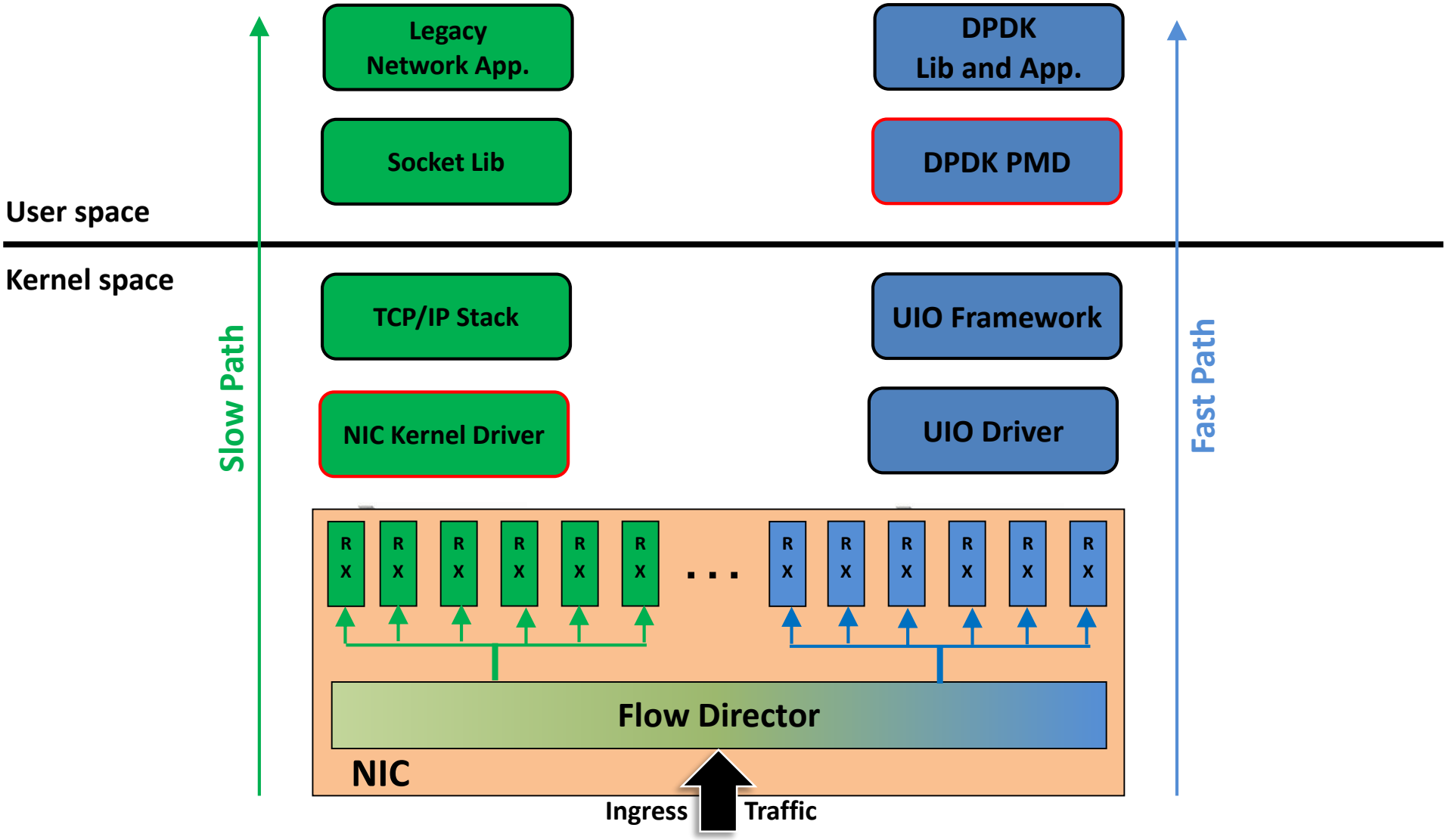


Kernel Bridging vs. L2Fwd Performance



Aggregated Performance at 64B small packet : 1.35 Mpps vs. 23.68Mpps

Motivation: What We Have & What To Build?



Bifurcated kernel driver would enable on-demand NIC resource partitioning while maintaining the high performance features

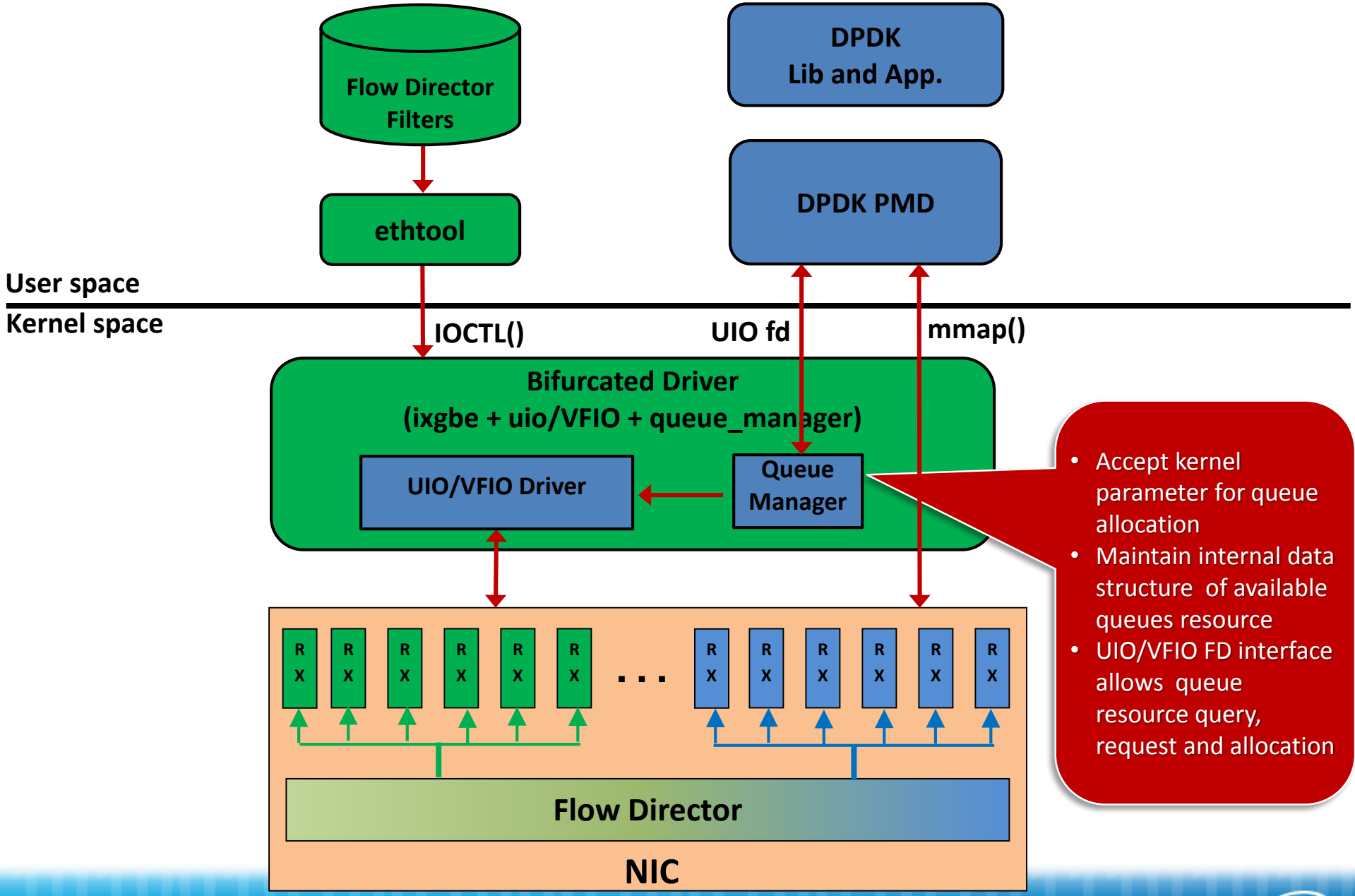
Design Principals

- **Loosely-coupled integration to minimize code change on both sides**
  - Two mechanism can work and evolve independently
  - Work together when needed based on agreed interface/protocols
  - The whole DPDK package is purely in user space

- **Master/salve mode**
  - Kernel driver as NIC master, DPDK PMD as NIC slave
  - Rx/Tx queue pair allocation and control via master
  - Slave only in charge of data-plane

- **NIC's flow director filters configuration only via ethtool**

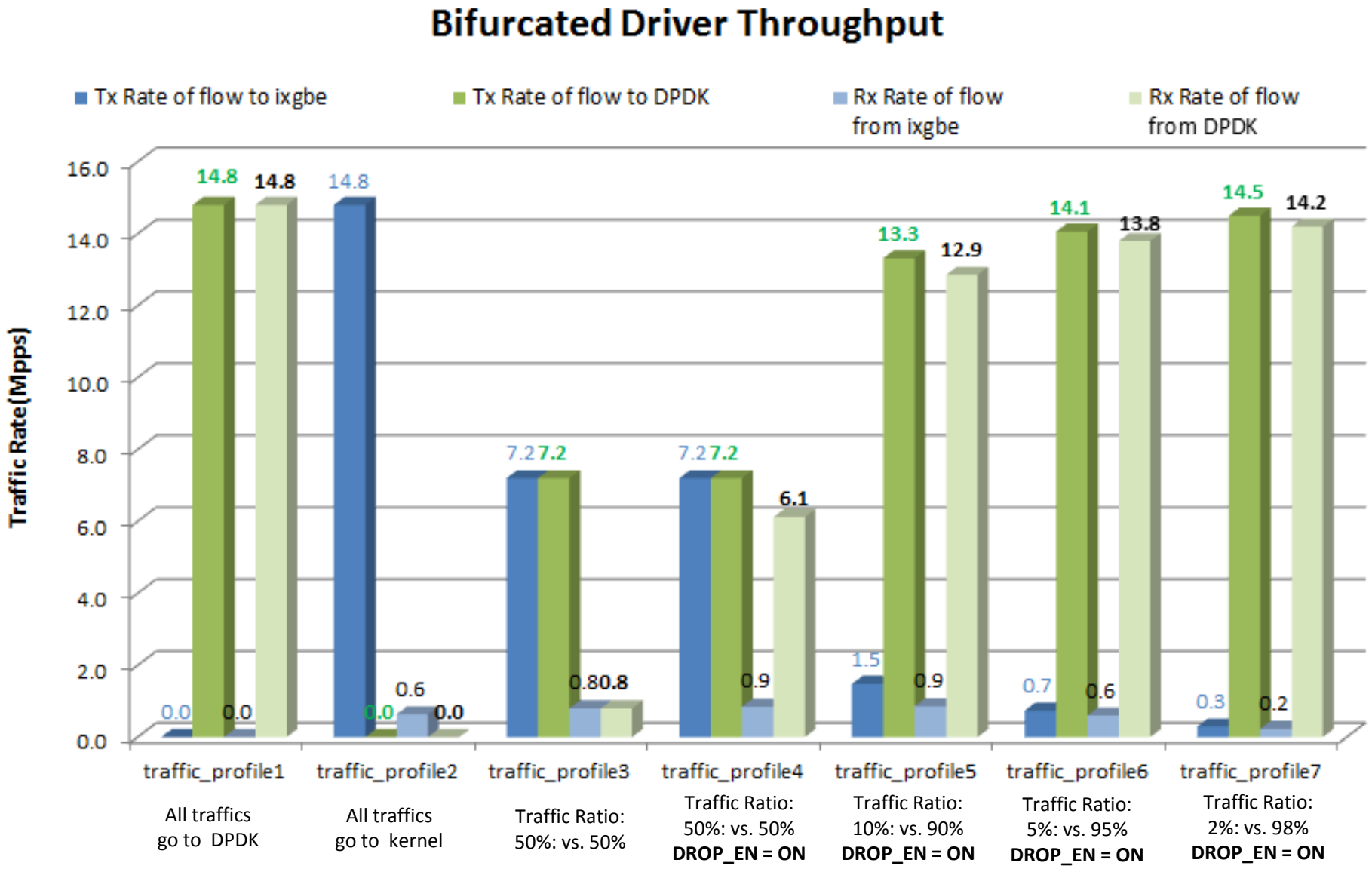
Software Architecture



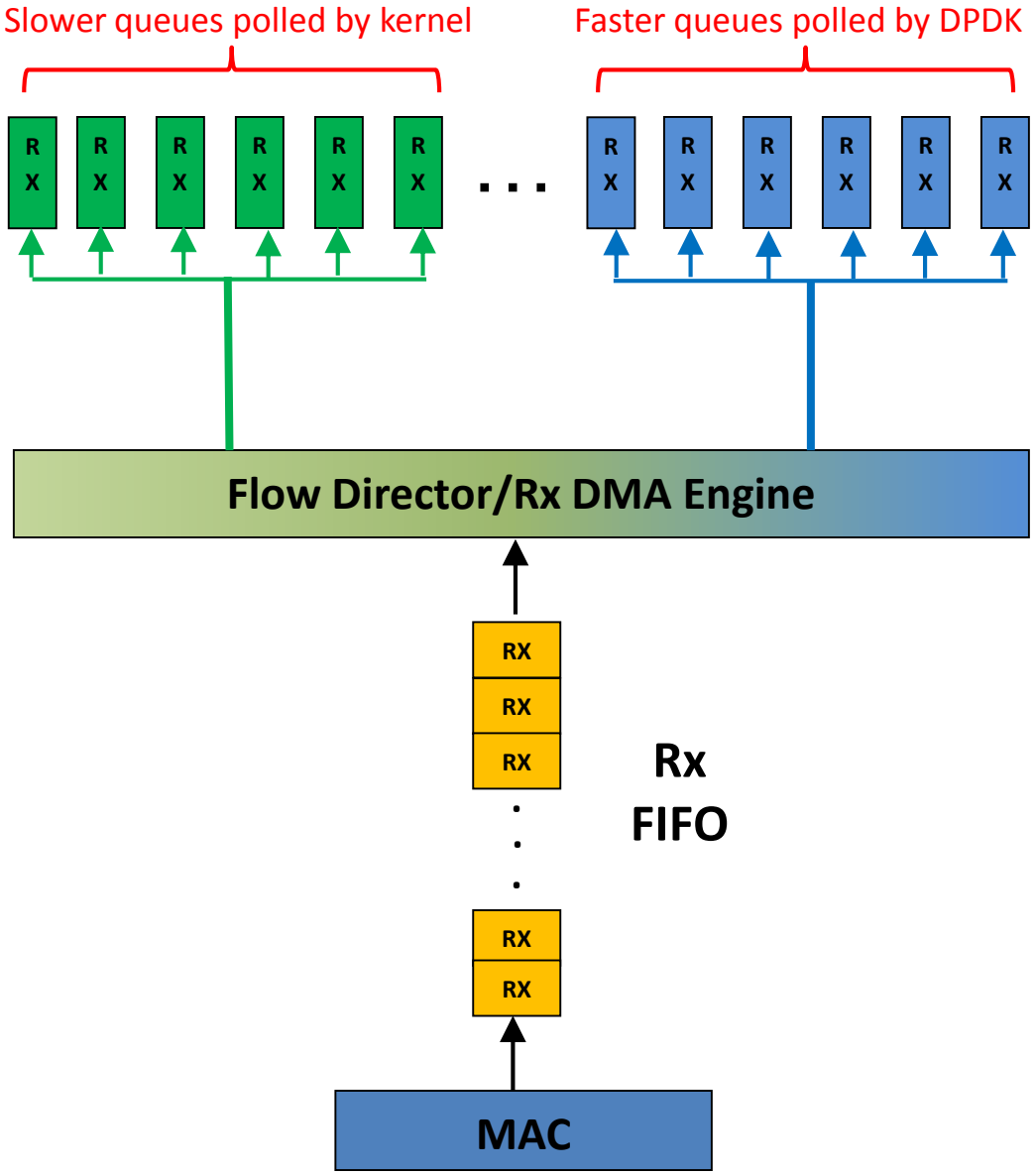
Startup Scripts

```
#  Set hugepage and load ixgbe_uio driver
        mount -t hugetlbfs nodev /mnt/huge
        modprobe uio
        insmod ixgbe_uio.ko  num_of_queue_pairs = 16

# Setup a Linux bridge connecting two Niantic ports
        brctl addbr br1
        brctl addif br1 p786p1
        brctl addif br1 p786p2
        brctl show br1
        ifconfig br1 up

# Enable and setup flow director rules
        ethtool -K p786p1 ntuple on   # enable flow director
        ethtool -N p786p1 flow-type udp4 src-ip 0.0.0.0 action 0   # direct flow to rxq 0  managed
                                                                     by ixgbe
        ethtool -N p786p1 flow-type udp4 src-ip 1.1.1.1 action 16   # direct flow to rxq 16
                                                                      managed by DPDK


# Start DPDK L2Fwd
      l2fwd -c 0x3 -n 4 --use-device=0000:08:00.0 --use-device=0000:08:00.1 -- -p 0x1
```

Performance Measurement



Why slow queue slow down fast queues?



- Rx FIFO "head of line blocking" in bifurcated configurations
- Can only move as fast as the quickest queues

- Solution: Enable SRRCTL.DROP_EN drops packets from Rx FIFO
  - Only drops packets when no buffers are available on Rx queue to DMA into
  - Allows faster rings to keep processing while slower rings drop packets

Bifurcated Driver Pros & Cons

Pros:

- Inherit DPDK' high performance gene
- DPDK is GPL free: no KNI & igb_uio any more
- DPDK no need to keep track of new NIC variants with different PCIE device ID
- Dynamically change the number of queues used for DPDK

Cons:

- Cross-dependency between ixgbe (or other NIC drivers) and DPDK
- DPDK can not control the NIC directly

Bifurcated Driver Upstream Patches

Patches for ixgbe pushed and accepted already, about to push for i40e and push fm10k in 2016

- The main use in today's driver is the use of Flow Director
- Made configuration and upstream acceptance easier
- Upstream patch need to be backport to the stand-alone versions of the drivers (the Soureforge versions)
- Giving "Bifurcated" a new name, Queue Splitting

UIO Bottom Interface to DPDK

- **Standard uio_pci_generic module included in the Linux kernel provides the uio capability**
- **For some devices which lack support for legacy interrupts, e.g. virtual function (VF) devices, the igb_uio module may be needed in place of uio_pci_generic.**

VFIO Bottom Interface to DPDK

- **In order to use VFIO, your kernel must support it. The VFIO kernel modules have been included in the Linux kernel since version 3.6.0 and are usually present by default.**

- **Also, to use VFIO, both kernel and BIOS must support and be configured to use IO virtualization (such as Intel® VT-d).**

Backup

Code Organization

- **/lib/librte_eal/linuxapp/ixgbe_uio**
  - Based on ixgbe version 3.18.7, added UIO support
  - No igb_uio needed any more, pci_unbind.py no longer needed
- **/lib/librte_pmd_ixgbe**
  - Only data-plane related driver functions are valid

```
static struct eth_dev_ops ixgbe_hbd_eth_dev_ops = {
        .dev_configure       = ixgbe_dev_configure,
        .dev_start          =  ixgbe_hbd_dev_start,
        .dev_stop           = ixgbe_hbd_dev_stop,
        .dev_close          = ixgbe_hbd_dev_close,
        .link_update         = ixgbe_hbd_dev_link_update,
        .dev_infos_get       = ixgbe_hbd_dev_info_get,
        .rx_queue_setup      = ixgbe_dev_rx_queue_setup,
        .rx_queue_release     = ixgbe_dev_rx_queue_release,
        .rx_queue_count      = ixgbe_dev_rx_queue_count,
        .rx_descriptor_done   = ixgbe_dev_rx_descriptor_done,
        .tx_queue_setup      = ixgbe_dev_tx_queue_setup,
        .tx_queue_release     = ixgbe_dev_tx_queue_release, };
```

  - Return error code if DPDK control-plane related function are invoked by application
  - NIC as well as Rx/Tx unit initialization disabled
  - Retrieve ixgbe initialized Rx/Tx queue pairs range

NIC Control can not be done DPDK anymore under bifurcated mode

Flow Director Filters

- To enable Flow Director
- ethtool -K ethX ntuple on
- To add a filter
- Use -U switch, Redhat supports the option, SUSE not.
- ethtool -U ethX flow-type tcp4 src-ip 168.0.0.1 action 1
- To Delete a filter
- Ethtool –U ethX delete N
- To show the list of filters:
- ethtool -u ethX
- Ethtool version for kernel 2.6.33 or later