DPDK
DATA PLANE DEVELOPMENT KIT

DPDK Summit

DPDK ARCHITECTURE AND ROADMAP DISCUSSION

KANNAN BABU RAMIA, INTEL
DEEPAK KUMAR JAIN, INTEL

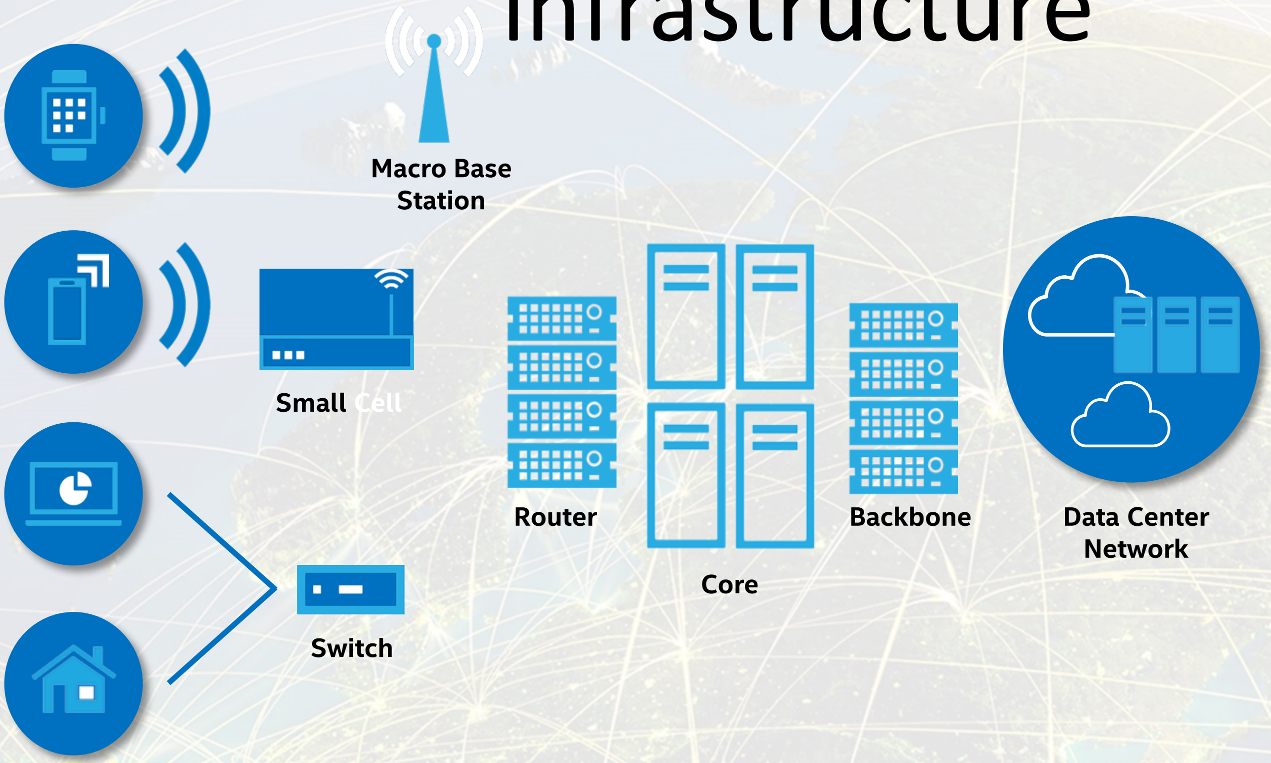
LEGAL DISCLAIMER

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm
- Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others.
-
- Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804
- Mileage may vary Disclaimer: Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **www.intel.com/benchmarks** Test and System Configurations: Estimates are based on internal Intel analysis using at least Data Plane Development Kit IpSec sample application on Intel(R) Xeon(R) CPU E5-2658 v4@ 2.30GHz with atleast using Intel(R) Communications Chipset(s) 8955 with Intel(R) QuickAssist Technology.

Agenda

- Key trends in network transformation
- DPDK role
- DPDK Architecture
- Multi Architecture/ Multi vendor support
- Open source projects using DPDK
- DPDK Roadmap
- Open Questions

End-to-end network infrastructure

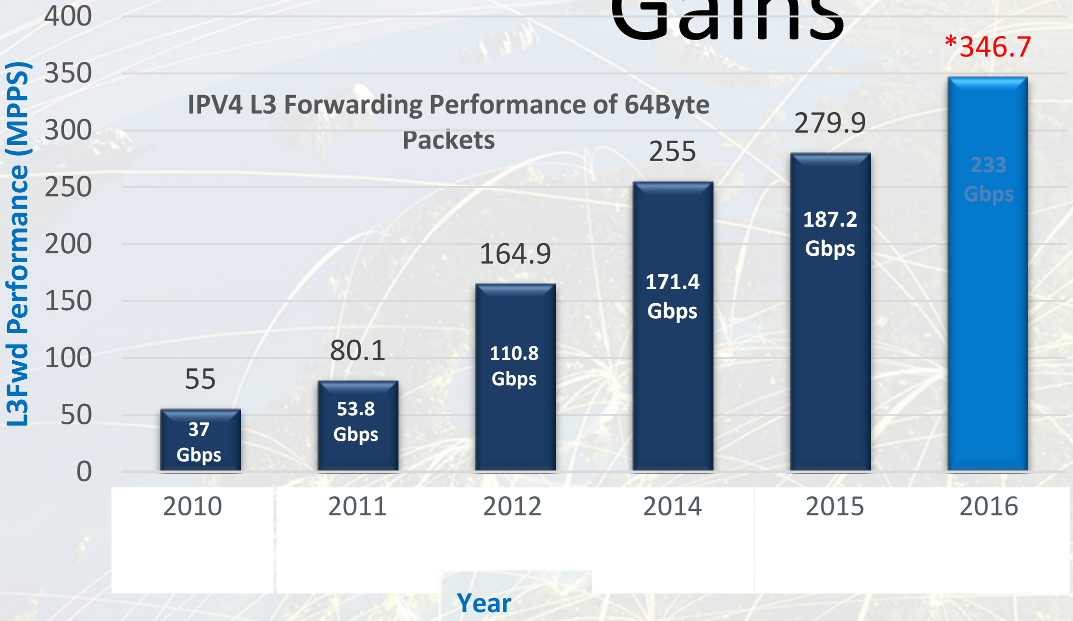


WORKLOAD CONVERGENCE

NETWORK VIRTUALIZATION

END-TO-END TRANSFORMATION

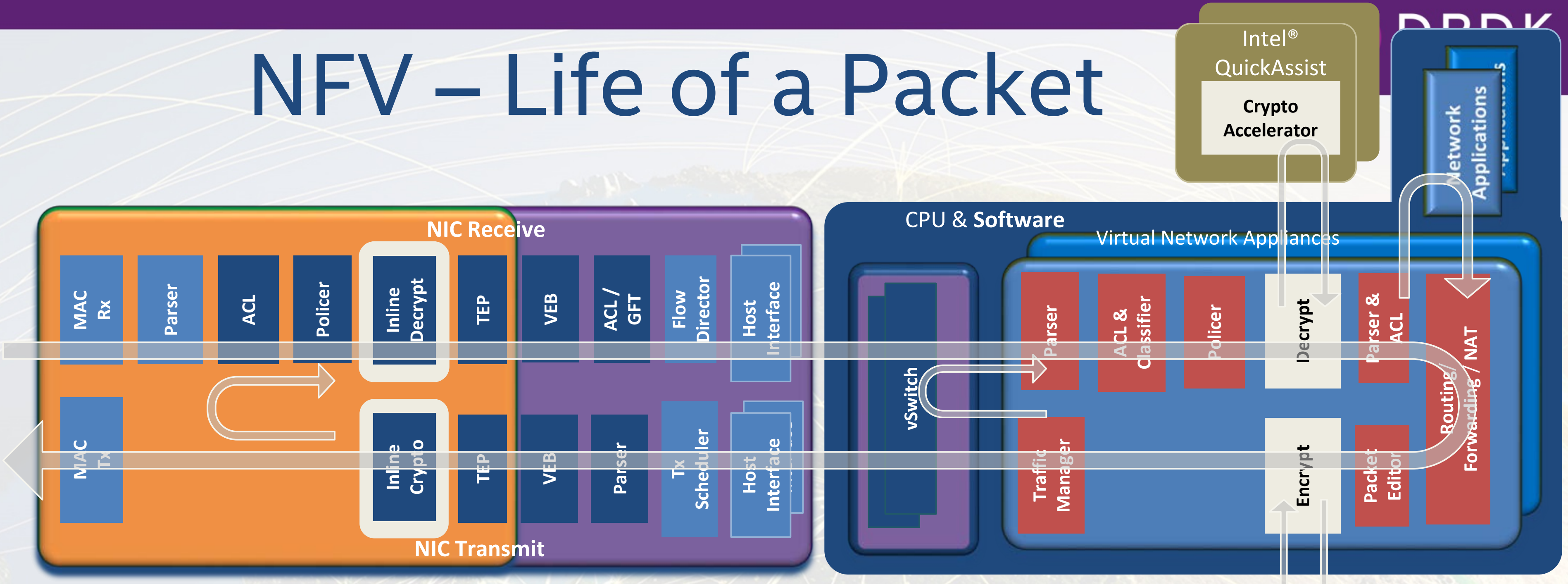
DPDK Generational Performance Gains

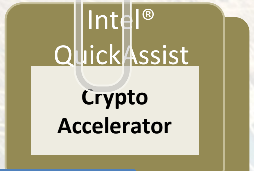


*System Configuration	
Hardware	
Platform	SuperMicro* - X10DRX
CPU	Intel® Xeon® E5-2658 v4 Processor
Chipset	Intel® C612 chipset
Sockets	2
Cores per Socket	14 (28 threads)
LL CACHE	30 MB
QPI/DMI	9.6GT/s
PCIe	Gen3x8
MEMORY	DDR4 2400 MHz, 1Rx4 8GB (total 64GB), 4 Channel per Socket
NIC	10 x Intel® Ethernet CNA XL710-QDA2PCI-Express Gen3 x8 Dual Port 40 GbE Ethernet NIC (1x40G/card)
NIC Mbps	40,000
BIOS	BIOS version: 1.0c (02/12/2015)
Software	
OS	Debian* 8.0
Kernel version	3.18.2
Other	DPDK 2.2.0

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.
Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those
assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit w

NFV – Life of a Packet

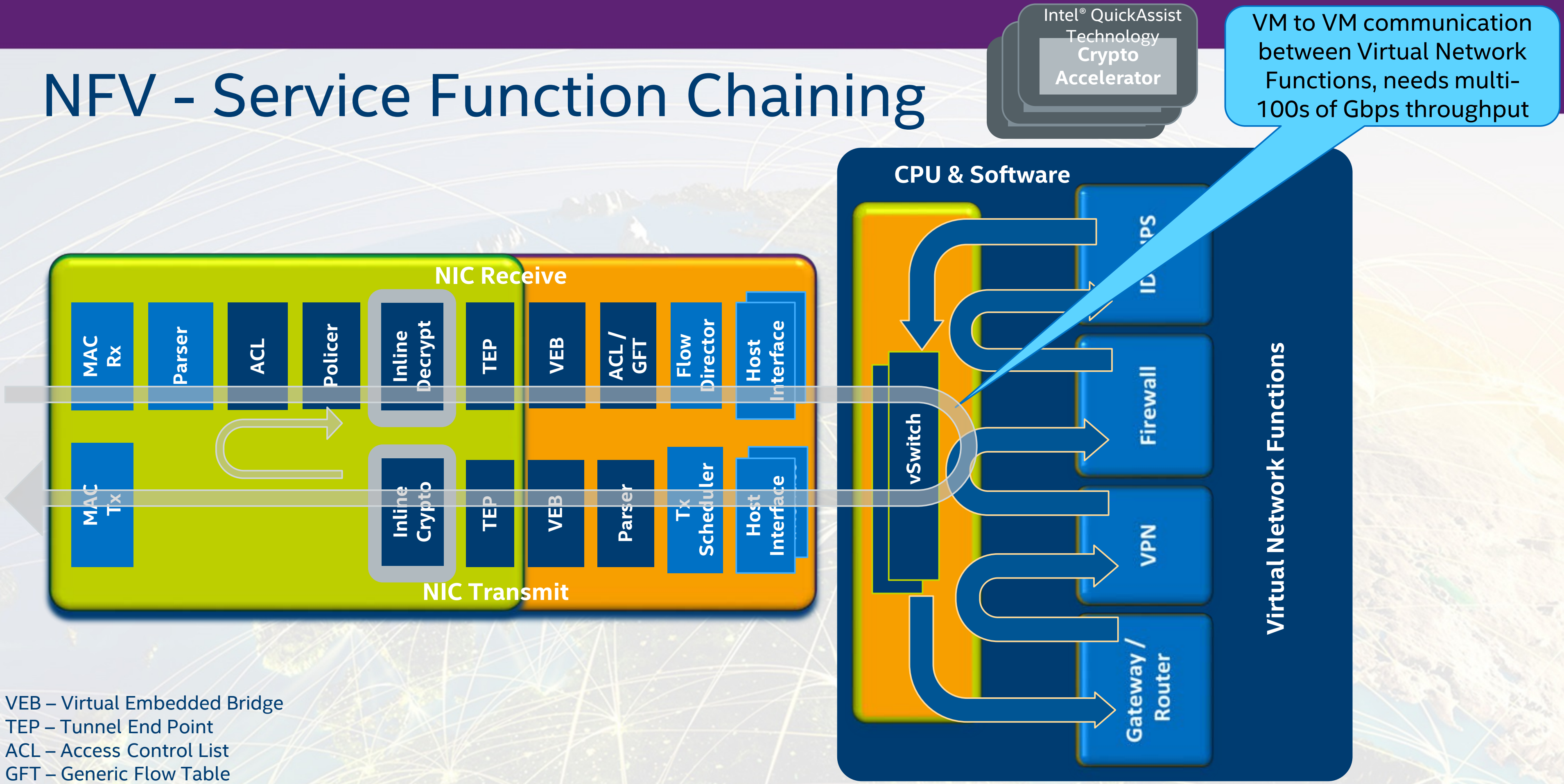




VEB – Virtual Embedded Bridge
TEP – Tunnel End Point
ACL – Access Control List
GFT – Generic Flow Table

Outer Header (Underlay)					Inner Payload (Overlay)					
Outer Ethernet Header	Outer IP Header	IPSec ESP transport	Outer UDP Header	Overlay Header (e.g. VXLAN)	Inner Ethernet Header	Inner IP Header	Inner L4 Header	Inner Data	IPSec ESP trailer	Outer CRC

NFV - Service Function Chaining

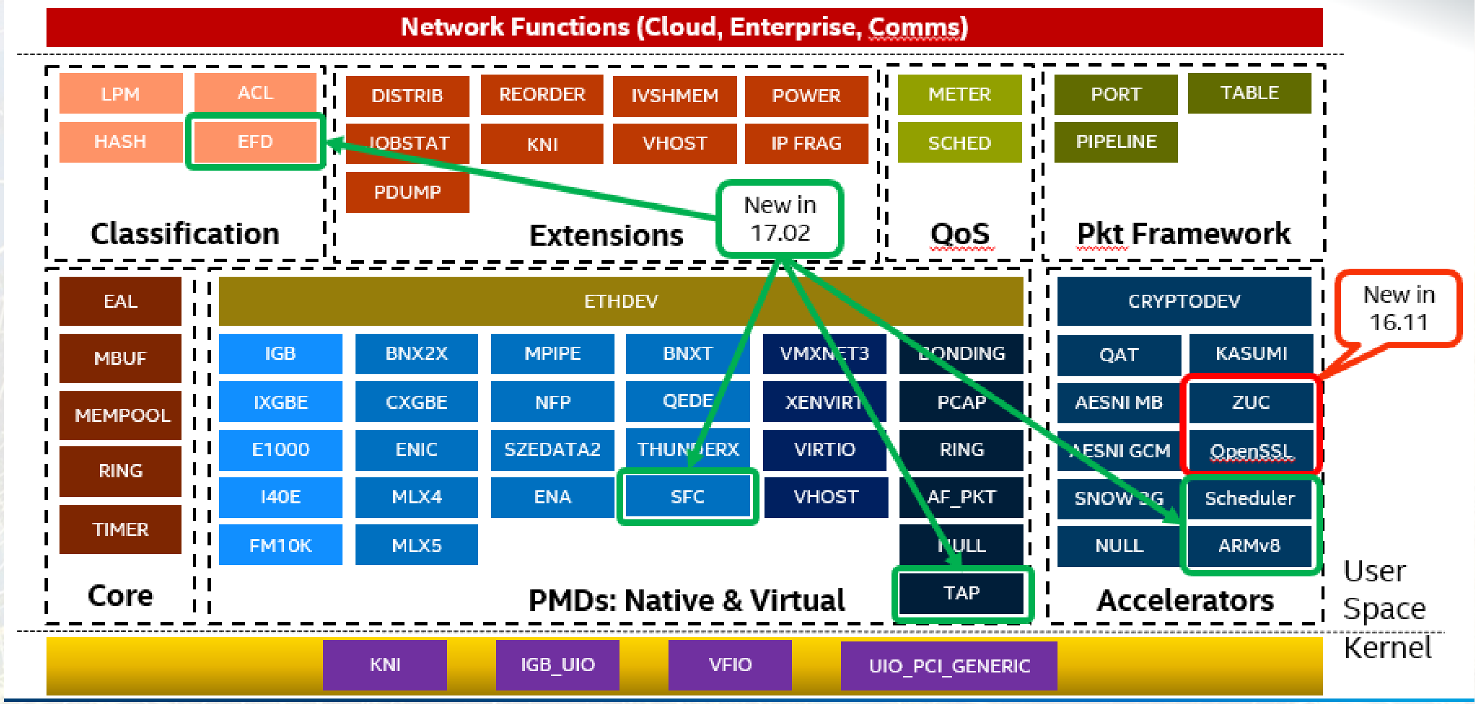


VEB – Virtual Embedded Bridge
TEP – Tunnel End Point
ACL – Access Control List
GFT – Generic Flow Table
VNF – Virtual Network Functions

DPDK Architecture

DPDK Fundamentals

- Implements run-to-completion and pipeline models
- No scheduler - all devices accessed by polling
- Supports 32-bit and 64-bit OSs with and without NUMA
- Scales from Intel® Atom™ to Intel® Xeon® processors
- Number of cores and processors is not limited
- Optimal packet allocation across DRAM channels
- Use of 2M & 1G hugepages and cache aligned structures
- Uses bulk concepts - processing 'n' packets simultaneously



Network Functions (Cloud, Enterprise, Comms)

Classification: LPM, ACL, HASH, EFD

Extensions: DISTRIB, REORDER, IVSHMEM, POWER, IOBSTAT, KNI, VHOST, IP FRAG, PDUMP

QoS: METER, SCHED

Pkt Framework: PORT, TABLE, PIPELINE

New in 17.02

Core: EAL, MBUF, MEMPOOL, RING, TIMER

PMDs: Native & Virtual: ETHDEV, IGB, BNX2X, MPIPE, BNXT, VMXNET3, BONDING, IXGBE, CXGBE, NFP, QEDE, XENVIRT, PCAP, E1000, ENIC, SZEDATA2, THUNDERX, VIRTIO, RING, I40E, MLX4, ENA, SFC, VHOST, AF_PKT, FM10K, MLX5, NULL, TAP

Accelerators: CRYPTODEV, QAT, KASUMI, AESNI MB, ZUC, AESNI GCM, OpenSSL, SNOW 3G, Scheduler, NULL, ARMv8

New in 16.11

User Space

Kernel

KNI, IGB_UIO, VFIO, UIO_PCI_GENERIC

DPDK
DATA PLANE DEVELOPMENT KIT

Multi-Architecture/ Multi-Vendor Support

CPU Architectures

POWER 8
IBM

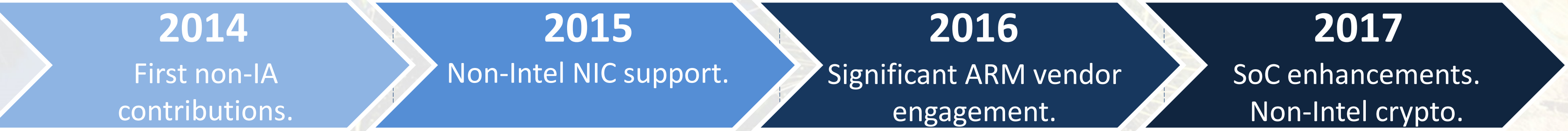
TILE-Gx
EZchip TECHNOLOGIES

ARM v7/v8
RehiveTech
CAVIUM

Enhanced ARM Support
CAVIUM
NXP
RehiveTech

Event API
CAVIUM

SoC Enhancements
NXP



2014
First non-IA contributions.

2015
Non-Intel NIC support.

2016
Significant ARM vendor engagement.

2017
SoC enhancements. Non-Intel crypto.

ENIC
CISCO

BNX2X
BROADCOM

MLX4/MLX5
Mellanox TECHNOLOGIES

SZEDATA2
liberouter

CXGBE
Chelsio Communications Accelerate

NFP
NETRONOME

ThunderX PMD
CAVIUM

QEDE
QLOGIC

BNXT
BROADCOM

ENA
SEMIHALF EMBEDDED SYSTEMS

ARMv8 Crypto
OcteonTX
LiquidIO
CAVIUM

DPAA2
NXP

AVP
WIND

SFC
SOLARFLARE

ARK
Atomic Rules

Poll Mode Drivers

DPDK CONSUMPTION

vSwitches

fd.io
VPP

OvS Open vSwitch

Lagopus

BESS

Packet Generators

TRex Realistic Traffic Generator

Pktgen

Ostinato

MoonGen

DPDK in OS Distros

redhat
Version 7.1 +

ubuntu
Version 15.10 +

FreeBSD
Version 10.1 +

WIND
Version 6 +

CentOS
Version 7.1 +

fedora
Version 22 +

Storage

SPDK

+ Many more

vRouters

OPENCONTRAIL

fd.io
VPP

CloudRouter™

TCP/IP Stacks

mTCP

fd.io
TLDK & VPP

Seastar

LWIP DPDK

DPDK Roadmap

Q1’17 (v 17.02)

- Added Elastic Flow Distributor library (rte_efd).L
- Added generic flow API (rte_flow).
- Added support for representing buses in EAL.
- Added APIs for MACsec offload support to the ixgbe PMD.
- Added VF Daemon (VFD) for i40e. – EXPERIMENTAL.
- virtio-user with vhost-kernel as another exceptional path.
- Added ARMv8 crypto PMD and updates to QAT, AESNI-MB PMDs.

Released

DPDK Roadmap

Q2’17 (v 17.05)

- **Added Eventdev PMD.**
- **Added event driven programming model library (rte_eventdev).**
- **Added bit-rate calculation, latency stats and information metric library.**
- **Kept consistent PMD batching behaviour.**
- **Added VFIO hotplug and vmxnet3 version 3 support.**
- **Added MTU feature support to Virtio and Vhost.**
- **Added interrupt mode support for virtio-user.**

In Development

DPDK Roadmap

Q2’17 (v 17.08)

- Generic QoS API
- Cryptodev Multi-Core SW Scheduler
- Generic Receive Offload
- Generic Flow Enhancements
- VF Port Reset for IXGBE
- API to Configure Queue Regions for RSS
- Support for IPFIX

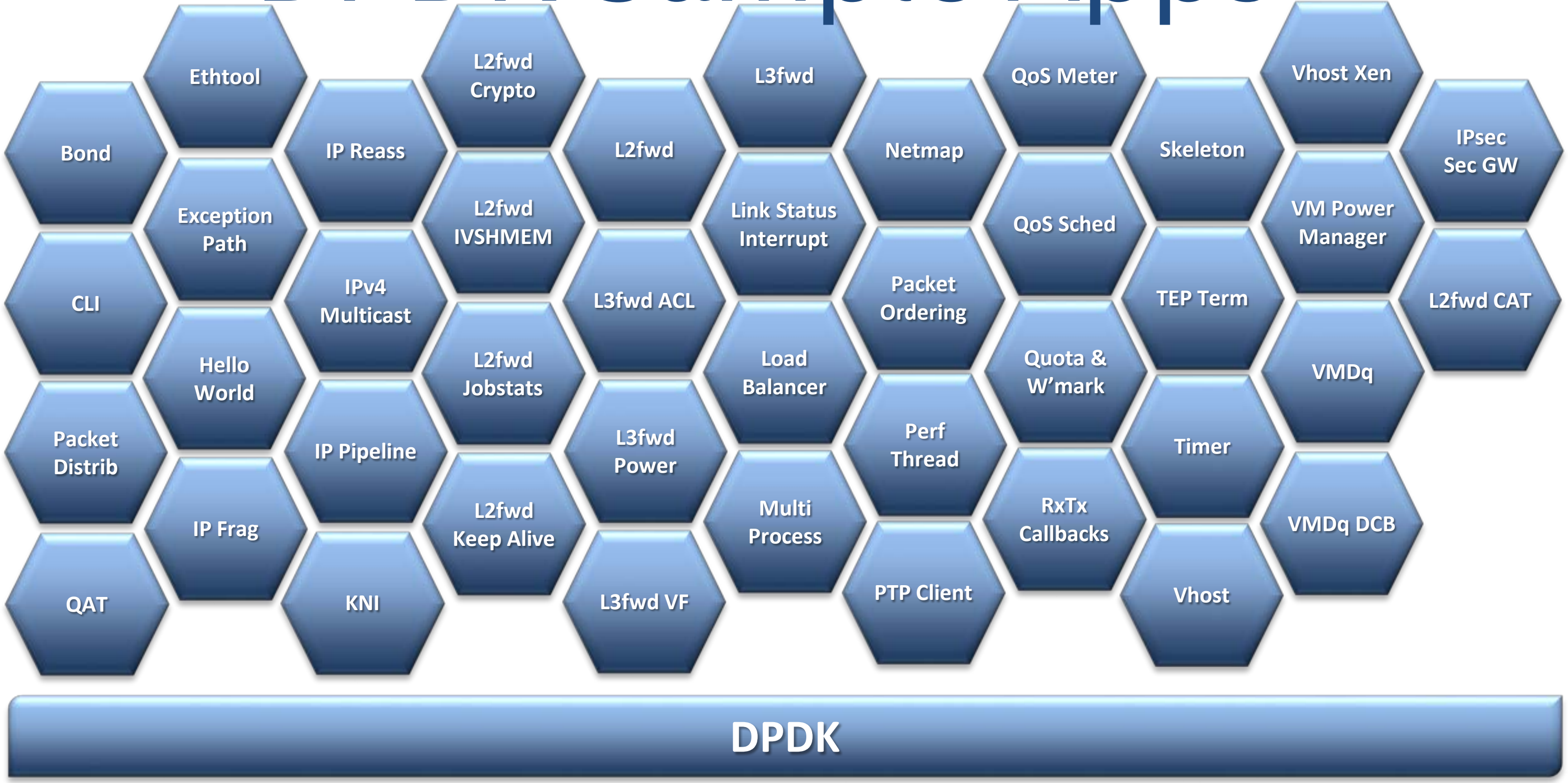
In Development

OPEN QUESTIONS?

- What is missing from DPDK?
- What are the major pain-points in using DPDK?
- What can be improved in DPDK? Build process? Logging?
- What are the big performance bottlenecks?
- Working with Kernel?

THANK YOU

DPDK Sample Apps



Bond

CLI

Packet Distrib

QAT

Ethtool

Exception Path

Hello World

IP Frag

IP Reass

IPv4 Multicast

IP Pipeline

KNI

L2fwd Crypto

L2fwd IVSHMEM

L2fwd Jobstats

L2fwd Keep Alive

L2fwd

L3fwd ACL

L3fwd Power

L3fwd VF

L3fwd

Link Status Interrupt

Load Balancer

Multi Process

Netmap

Packet Ordering

Perf Thread

PTP Client

QoS Meter

QoS Sched

Quota & W’mark

RxTx Callbacks

Skeleton

TEP Term

Timer

Vhost

Vhost Xen

VM Power Manager

VMDq

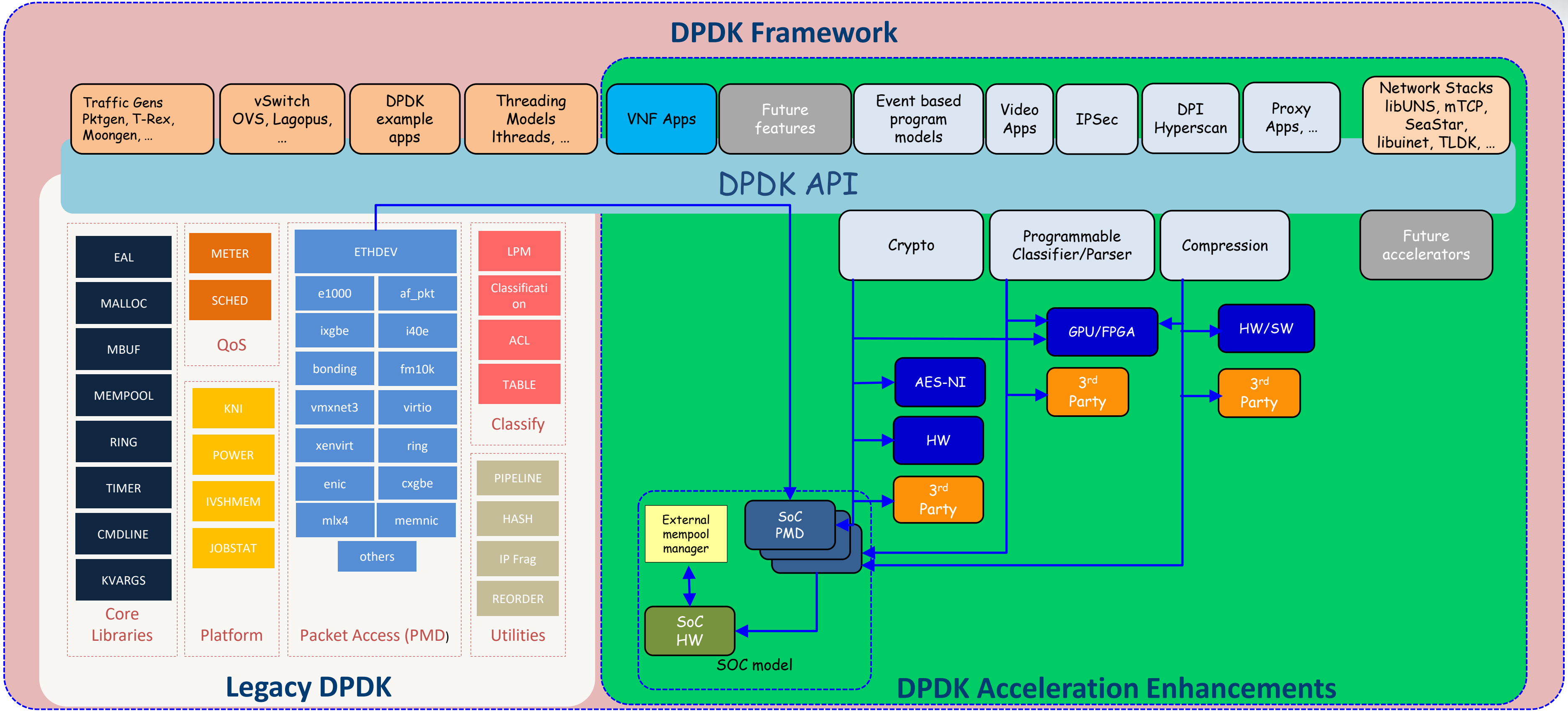
VMDq DCB

IPsec Sec GW

L2fwd CAT

DPDK

DPDK Acceleration Enhancements



DPDK Framework

Traffic Gens Pktgen, T-Rex, Moongen, …

vSwitch OVS, Lagopus, …

DPDK example apps

Threading Models lthreads, …

VNF Apps

Future features

Event based program models

Video Apps

IPSec

DPI Hyperscan

Proxy Apps, …

Network Stacks libUNS, mTCP, SeaStar, libuinet, TLDK, …

DPDK API

Core Libraries: EAL, MALLOC, MBUF, MEMPOOL, RING, TIMER, CMDLINE, KVARGS

QoS: METER, SCHED

Platform: KNI, POWER, IVSHMEM, JOBSTAT

Packet Access (PMD): ETHDEV, e1000, af_pkt, ixgbe, i40e, bonding, fm10k, vmxnet3, virtio, xenvirt, ring, enic, cxgbe, mlx4, memnic, others

Classify: LPM, Classification, ACL, TABLE

Utilities: PIPELINE, HASH, IP Frag, REORDER

Legacy DPDK

Crypto

Programmable Classifier/Parser

Compression

Future accelerators

GPU/FPGA

HW/SW

AES-NI

HW

3rd Party

3rd Party

3rd Party

External mempool manager

SoC PMD

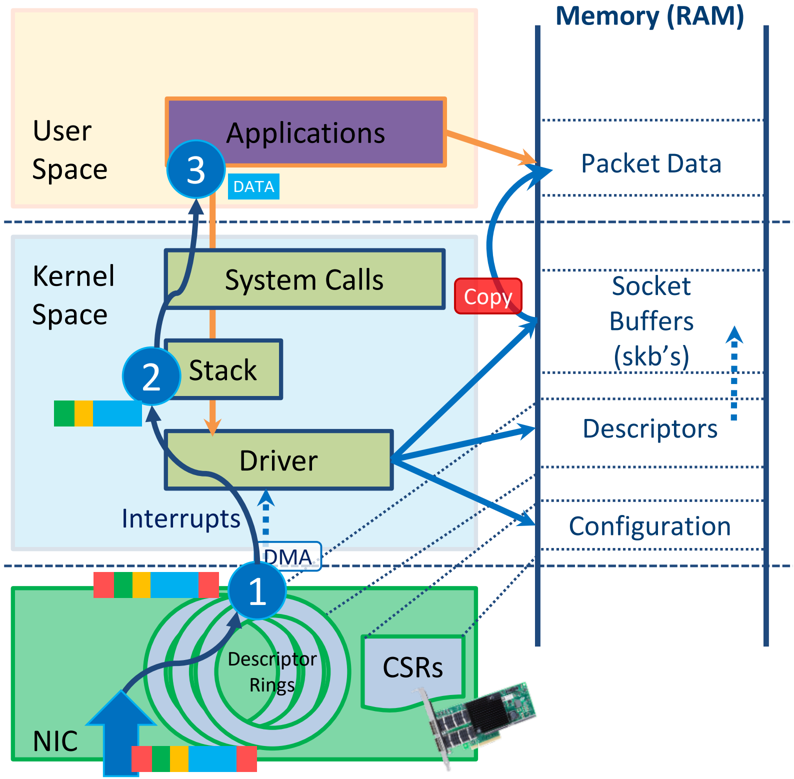
SoC HW

SOC model

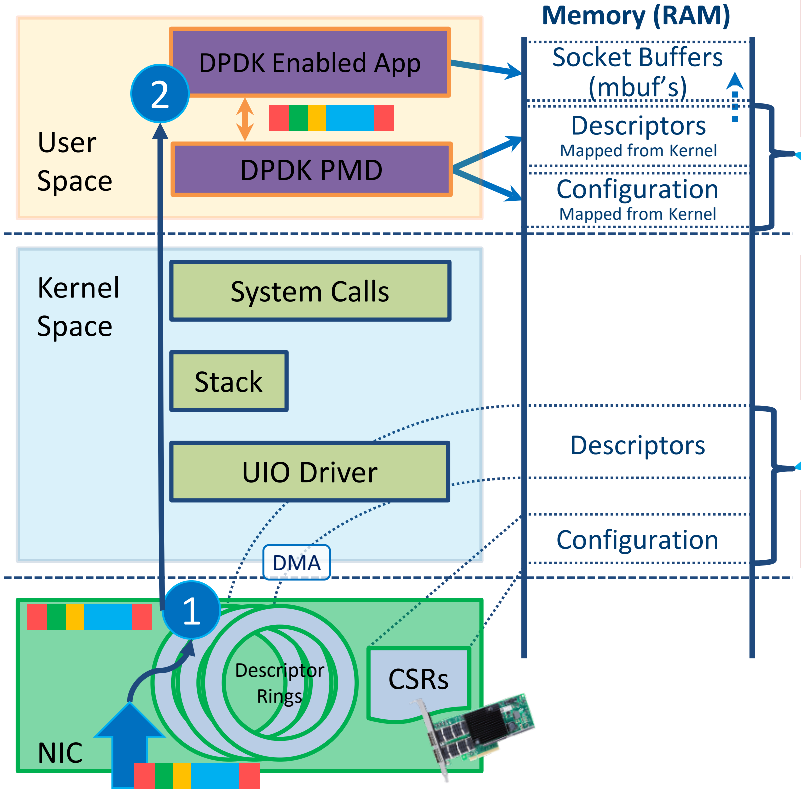
DPDK Acceleration Enhancements

Packet Processing Kernel vs. User Space

Kernel Space Driver



User Space Driver with Zero Copy



Benefit #3
Network stack can be streamlined and optimized

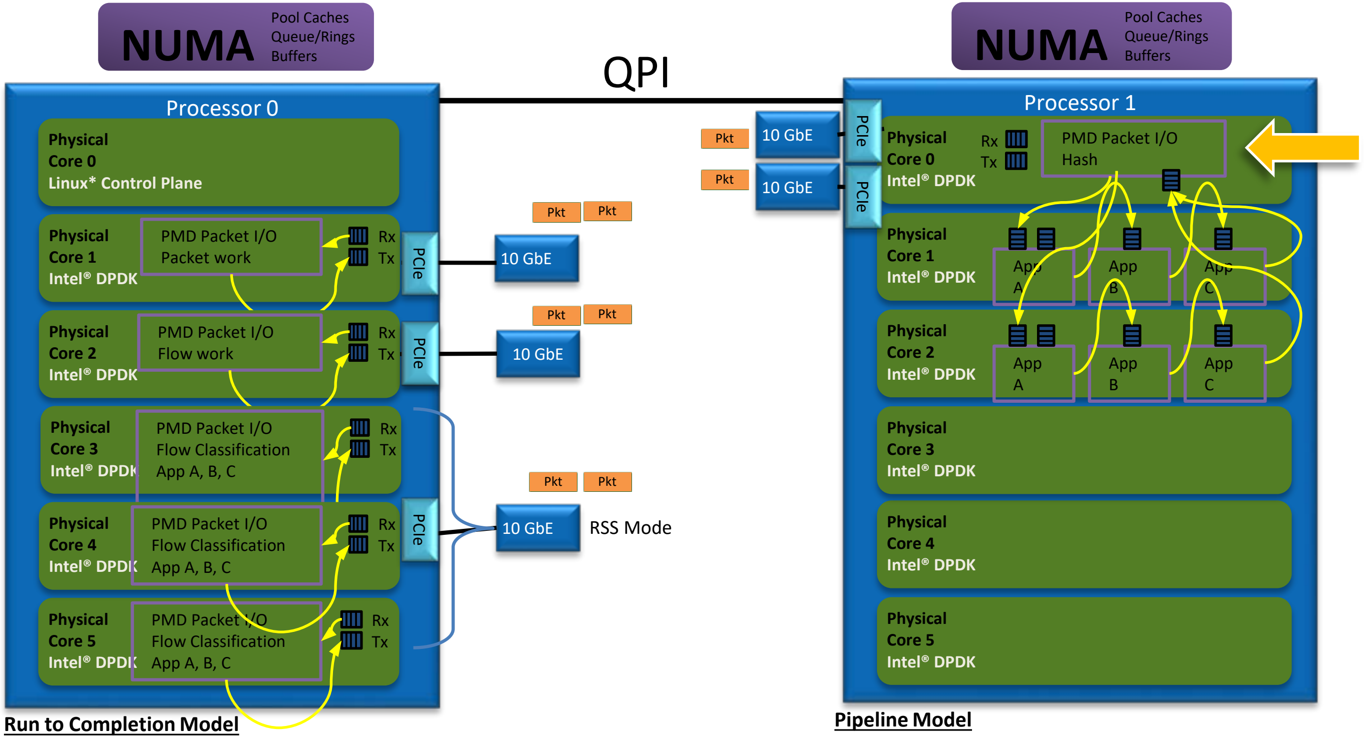
Benefit #1
Removed Data copy from Kernel to User Space

Benefit #2
No Interrupts

DPDK IN-DEPTH

PCIe* Connectivity and Core Usage

Using run-to-completion or pipeline software models



Can handle more I/O on fewer cores with vectorization

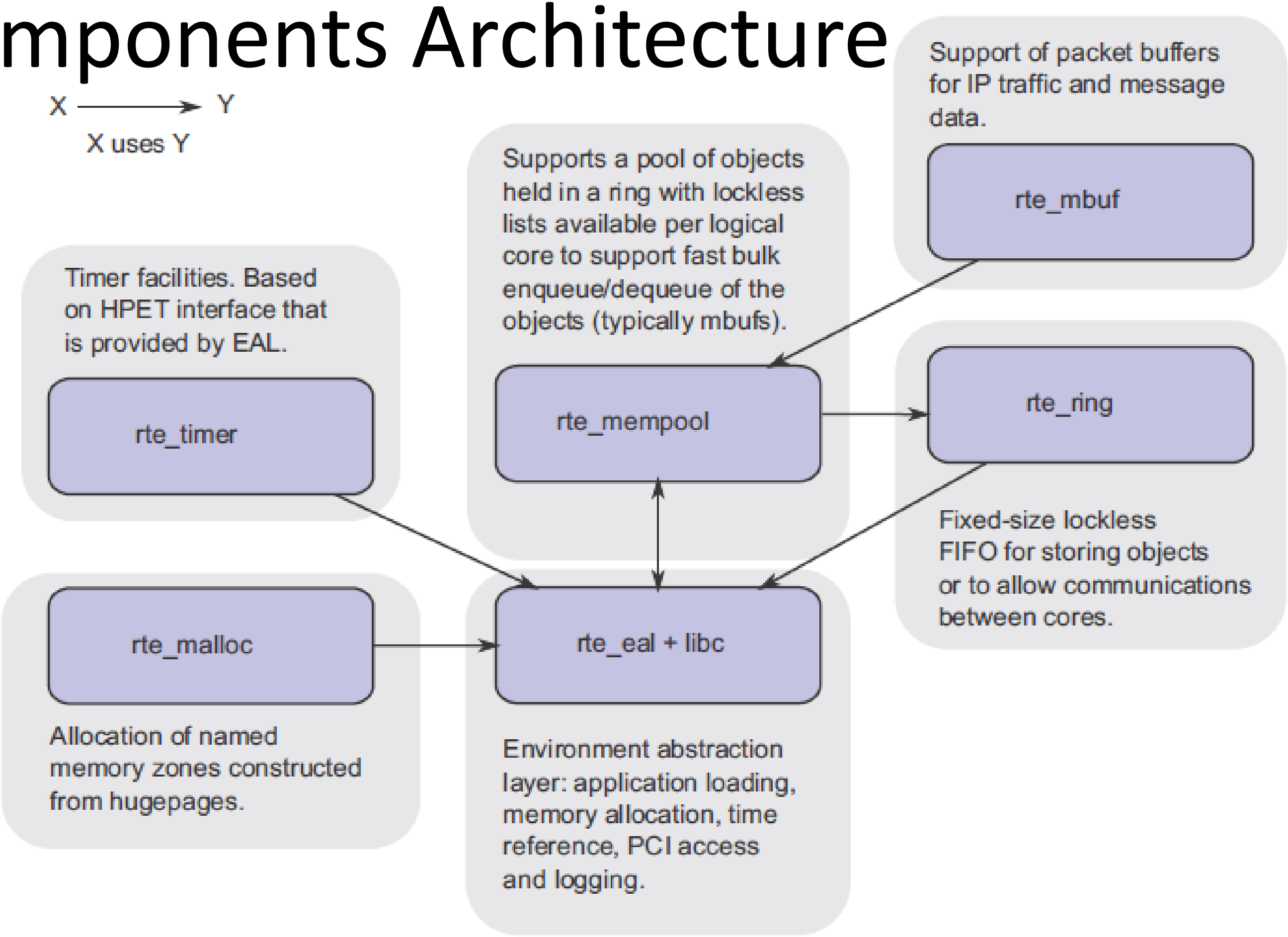
Run to Completion Model

- I/O and Application workload can be handled on a single core
- I/O can be scaled over multiple cores

Pipeline Model

- I/O application disperses packets to other cores
- Application work performed on other cores

Core Components Architecture



Support of packet buffers for IP traffic and message data.

Supports a pool of objects held in a ring with lockless lists available per logical core to support fast bulk enqueue/dequeue of the objects (typically mbufs).

Timer facilities. Based on HPET interface that is provided by EAL.

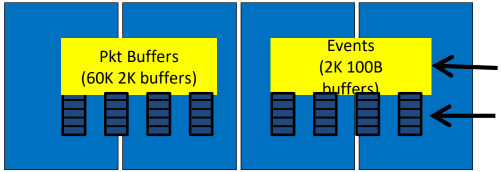
Fixed-size lockless FIFO for storing objects or to allow communications between cores.

Allocation of named memory zones constructed from hugepages.

Environment abstraction layer: application loading, memory allocation, time reference, PCI access and logging.

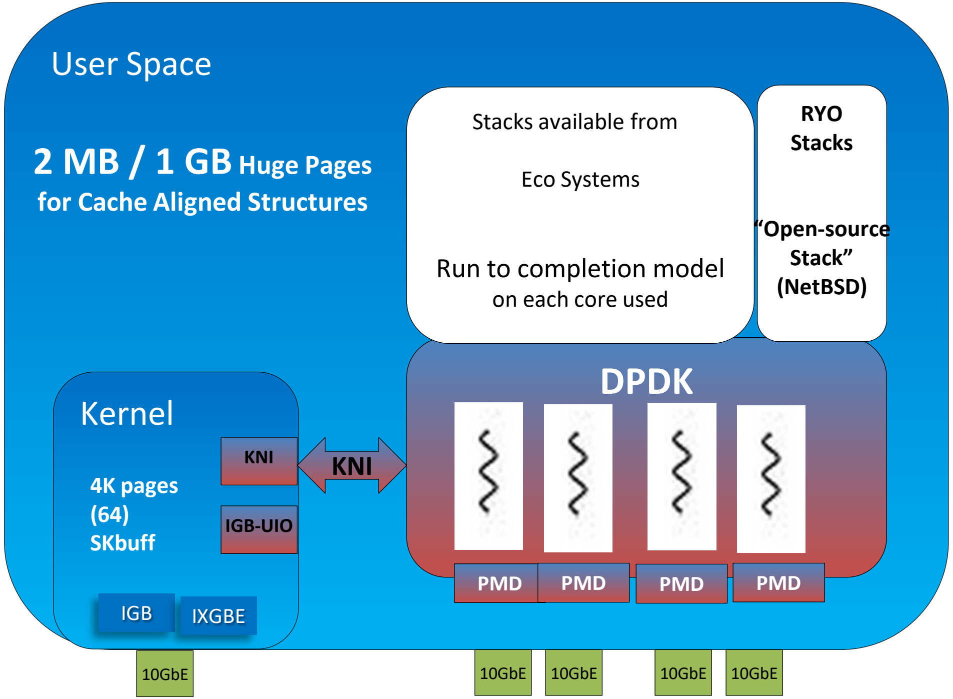
DPDK model

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



Rings for cached buffers

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



User Space

2 MB / 1 GB Huge Pages
for Cache Aligned Structures

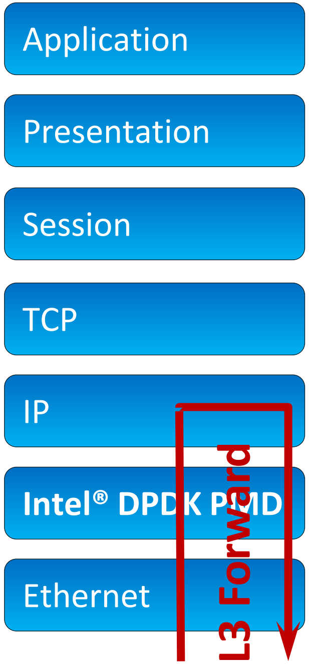
Stacks available from

Eco Systems

Run to completion model
on each core used

RYO Stacks

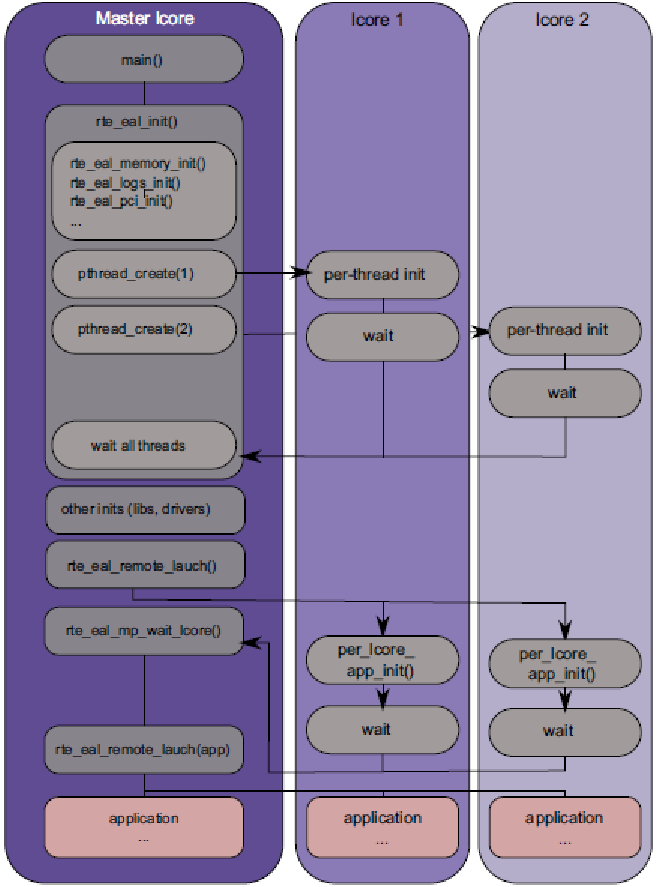
"Open-source Stack" (NetBSD)



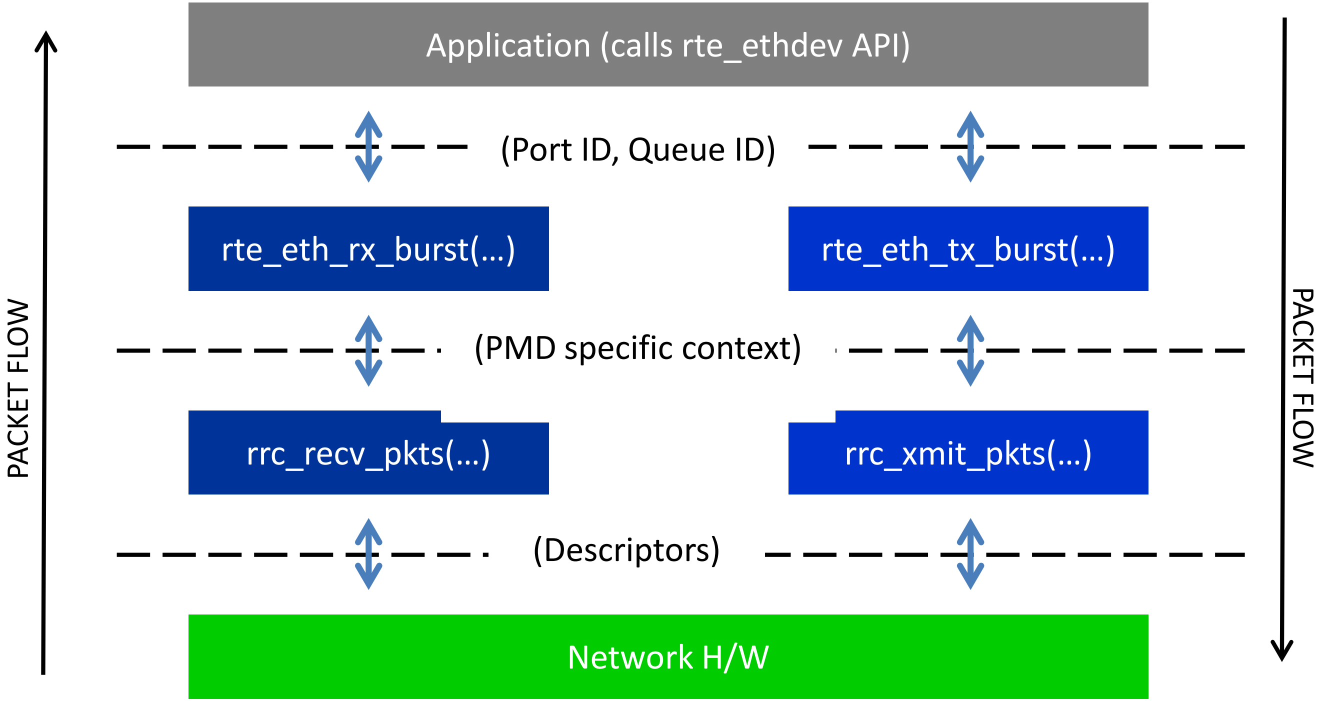
High Performance Components of DPDK

- Environment Abstraction Layer
  - Abstracts huge-page file system, provides multi-thread and multi-process support, etc.
- Memory Manager
  - Responsible for allocating pools of objects in memory. A pool is created in huge page memory space and uses a ring to store free objects. It also provides an alignment helper to ensure that objects are padded to spread them equally on all DRAM channels.
- Buffer Manager
  - Reduces by a significant amount the time the operating system spends allocating and de-allocating buffers. The Intel® DPDK pre-allocates fixed size buffers which are stored in memory pools.
- Queue Manager
  - Implements safe lockless queues, instead of using spinlocks, that allow different software components to process packets, while avoiding unnecessary wait times.
- Flow Classification
  - Provides an efficient mechanism which incorporates Intel® Streaming SIMD Extensions (Intel® SSE) to produce a hash based on tuple information so that packets may be placed into flows quickly for processing, thus greatly improving throughput.

EAL Initialization in a Linux Environment



Ethernet Device Framework



Application (calls rte_ethdev API)

(Port ID, Queue ID)

rte_eth_rx_burst(…)

rte_eth_tx_burst(…)

(PMD specific context)

rrc_recv_pkts(…)

rrc_xmit_pkts(…)

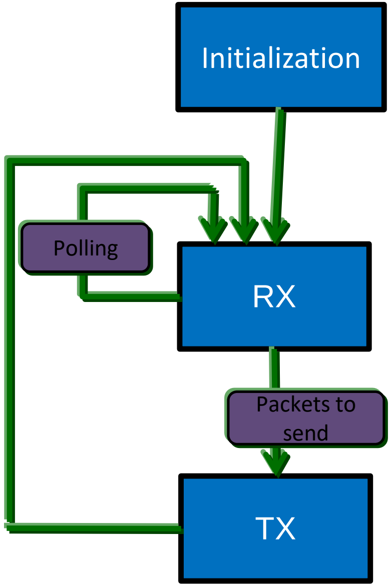
(Descriptors)

Network H/W

PACKET FLOW

PACKET FLOW

30,000 ft overview of packet flow



1. Initialization
   - Init Memory Zones and Pools
   - Init Devices and Device Queues
   - Start Packet Forwarding Application
2. Packet Reception (RX)
   - Poll Devices’ RX queues and receive packets in bursts
   - Allocate new RX buffers from per queue memory pools to stuff into descriptors
3. Packet Transmission (TX)
   - Transmit the received packets from RX
   - Free the buffers that we used to store the packets