



**OPEN**  
Compute Project



**OCP U.S. SUMMIT 2017**

Santa Clara, CA



# SmartNIC Data Plane Acceleration & Reconfiguration

Nic Viljoen, Senior Software Engineer, Netronome

OPEN HARDWARE.

OPEN SOFTWARE.

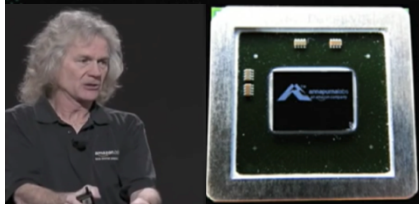
OPEN FUTURE.





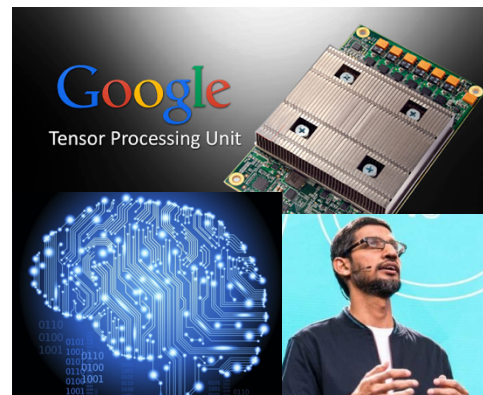
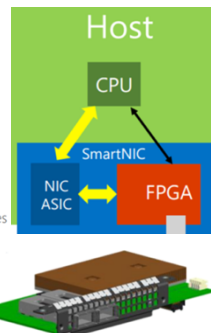
## 2016 CUSTOM SILICON

- Custom Si & 25GbE
  - 2x 25GbE cheaper & higher bandwidth than 40GbE
- Amazon Annapurna ASIC
- Second generation Enhanced Networking
- AWS controls silicon, hardware & software
- AWS pace of innovation



## Azure SmartNIC

- Use an FPGA for reconfigurable functions
  - FPGAs are already used in Bing (Catapult)
  - Roll out Hardware as we do software
- Programmed using Generic Flow Tables (GFT)
  - Language for programming SDN to hardware
  - Uses connections and structured actions as primitives
- SmartNIC can also do Crypto, storage acceleration, and more



Large R&D budgets, deep acceleration software expertise  
Proprietary silicon and hardware-based acceleration

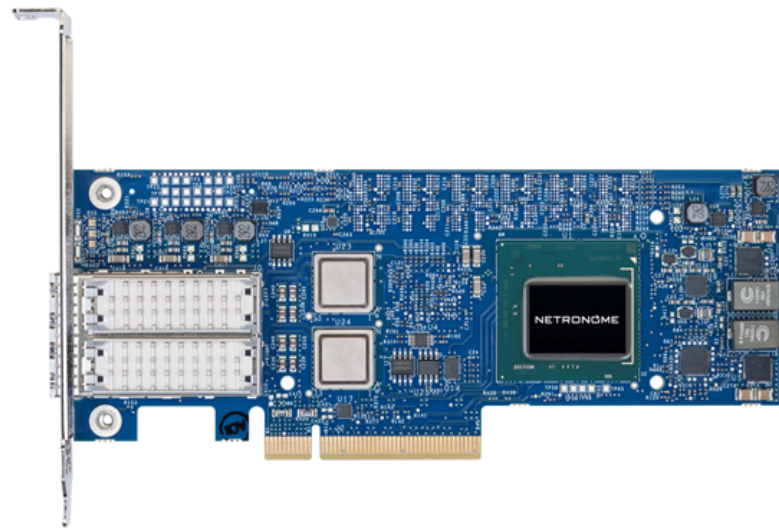
Rest of the market deploying cloud technologies need off-the-shelf solutions

TCAM, FPGA devices in NICs

GP CPUs – ARM/MIPS in NICs

## Network Flow Processor (NFP)

- ▶ Programmable, run-to-completion
- ▶ Performant, highly multi-threaded
- ▶ NFP 4000/6000 silicon family
- ▶ Up to 120 processing cores
- ▶ Hardware accelerators

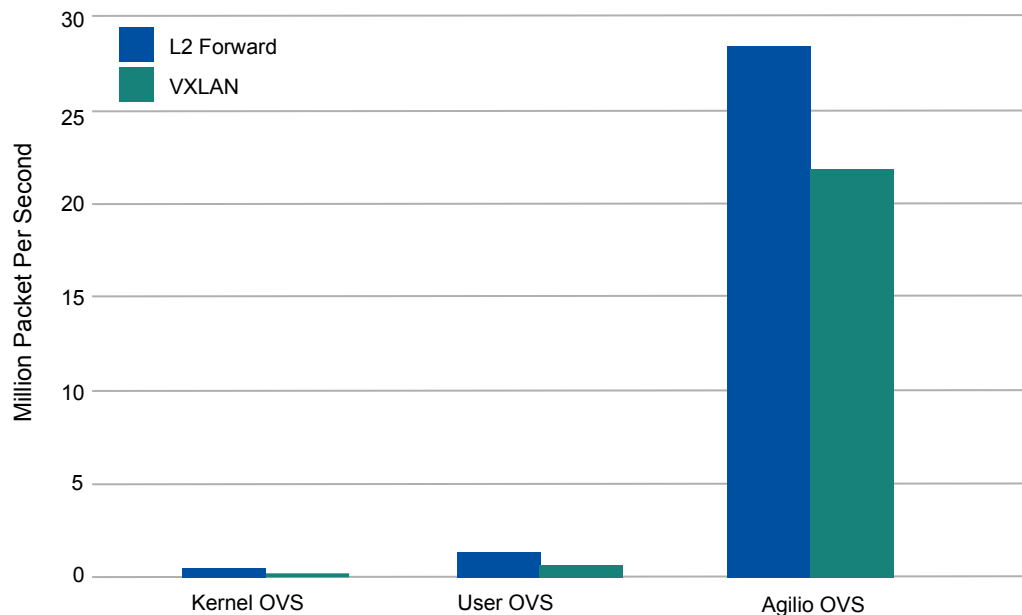


**Agilio® CX Family of SmartNICs**

Based on NFP-4000/6000

Low profile PCIe adapters consume  
<25W, enable up to 50GbE throughput  
and saves as many as 12 x86 CPU cores

## Throughput with single server CPU core



**50X**

Efficiency Gain vs. Kernel OVS

**20X**

Efficiency Gain vs. User OVS

Using Netronome Agilio CX  
SmartNICs (2x 10/25/40GbE)  
(PCIe Gen3, Low Profile, <25W)

[https://www.netronome.com/media/redactor\\_files/WP\\_OVS\\_Benchmarking.pdf](https://www.netronome.com/media/redactor_files/WP_OVS_Benchmarking.pdf)

## NFV Infrastructure



## Cloud Networking



## Security



Deployment at scale needs a new level of interaction between firmware and software

- ▶ Fixed Function NIC + Host Driver may be insufficient for these requirements
- ▶ Need new application development and deployment mechanisms
- ▶ Need community based, stable, maintainable and flexible

Mass adoption of SmartNICs requires easy DevOps model

## Transparent bytecode acceleration using BPF

- ▶ Highly flexible offload, performant, low OPEX, upstream
- ▶ Code for kernel based acceleration can be repurposed

## Transparent offload of networking applications (e.g. OVS, KTLS)

- ▶ Reliable, performant, low OPEX
- ▶ Ties consumer to application

## Fully custom targeted application built using SDK

- ▶ Highly flexible offload, optimal performance
- ▶ High OPEX, large teams

## Upstream verifier and JIT

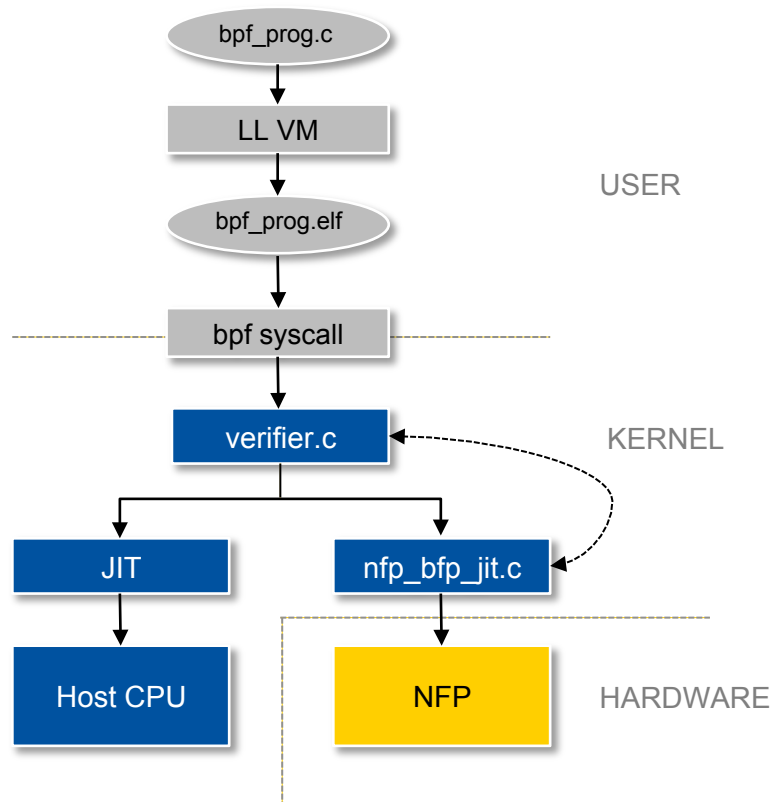
- ▶ Ensures maintainability and stability
- ▶ Solving the 'whack-a-mole' problem

## Bytecode offload – Hardware transparency

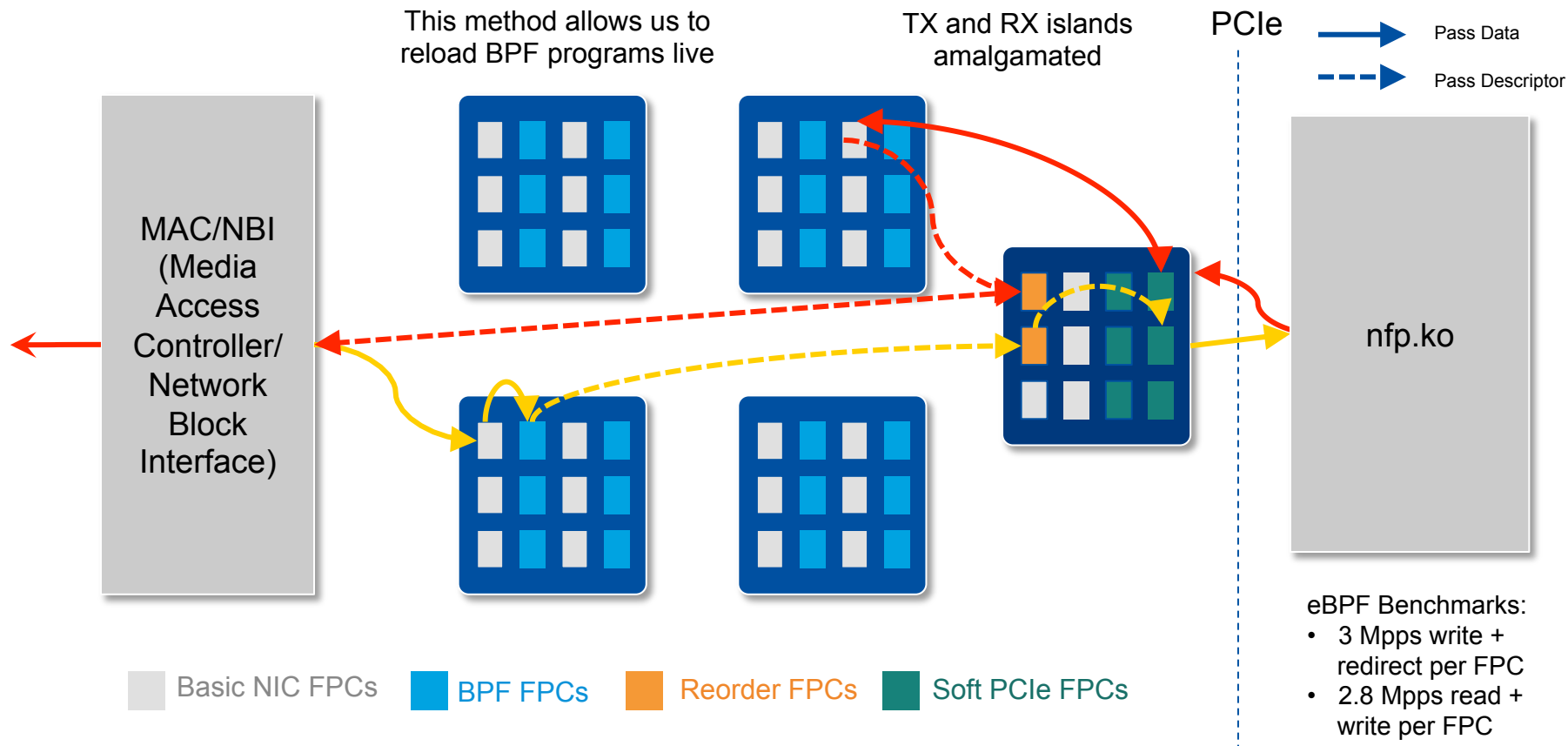
- ▶ XDP or TC acceleration with no custom code

## Bytecode offload – Future Proof

- ▶ Future support e.g. nftables bytecode



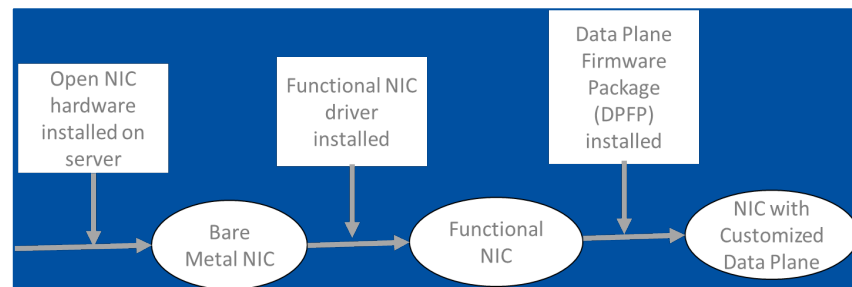




- eBPF Benchmarks:
- 3 Mpps write + redirect per FPC
  - 2.8 Mpps read + write per FPC

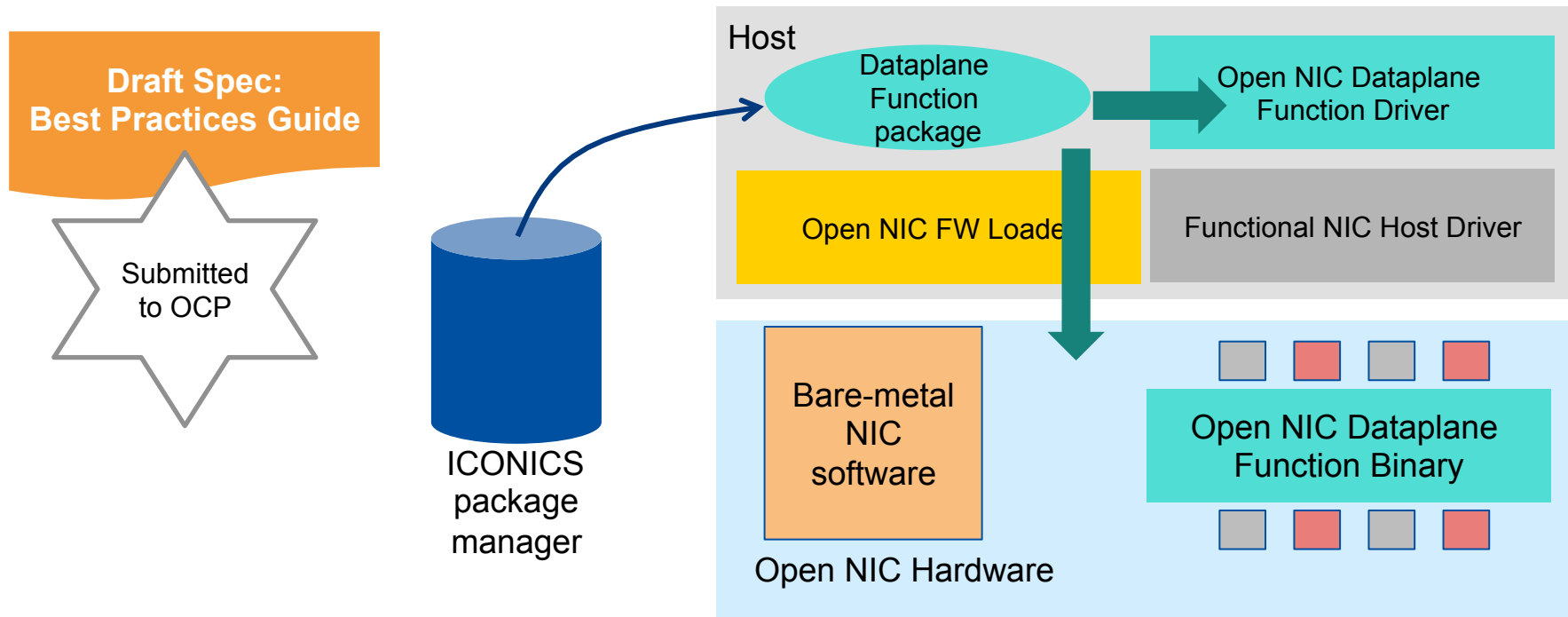
## ICONICS: Installation and Configuration of Open NICs

- ▶ Support “bare-metal NIC” mode
  - Known as Open NIC
- ▶ Open NIC data plane is configurable on-demand
- ▶ Open NIC will be customizable in the field
- ▶ No-touch conflict-free memoryless updates
- ▶ Integrated with current processes
- ▶ Utilize Open and Common Source Update Tools
- ▶ Secure updates from diverse sources



**Open NIC Operating Model**

ICONICS Draft Specification: Best practices for managing SmartNICs



Use specification best practices & app repository to manage bare metal NIC personalization at scale

SmartNICs add significant flexibility and reduce TCO of data centers

- ▶ Shown publicly by hyperscalers e.g. Amazon, Microsoft etc.

Infrastructure stability and maintainability requires incremental updates of firmware

Community based approach to SmartNIC DevOps leads to faster adoption

- ▶ Process has started in kernel community e.g. `nfp_bpf_jit`
- ▶ ICONICS guide

Visit Netronome @ Booth A14

- ▶ BPF -- Early prototype of eBPF Data Plane
- ▶ ICONICS – On-demand reconfiguration of SmartNIC Data Plane

Visit Netronome @ Booth A14 to learn more about SmartNICs

## Netdev talk on transparent eBPF offload

- ▶ <https://youtu.be/-5BzT1ch19s>

## eBPF offload demo

- ▶ <https://youtu.be/apU5sg0Ui5U?t=2003>

## ICONICS: Draft Specification and upcoming apps repository

- ▶ [www.iconics.io](http://www.iconics.io) and associated GitHub repositories

## Open-NFP: Resources, academic projects, webinars

- ▶ <http://open-nfp.org>

## Get your hands on SmartNICs: 2x10G, 2x25G, 1x40G, 2x40G, 1x100G

- ▶ Commercial Use: [Colfax Direct](#)
- ▶ Academic: <http://open-nfp.org/resources/>



# OPEN

Compute Project





**Compute Servers** for NFV Infrastructure and Cloud Networking

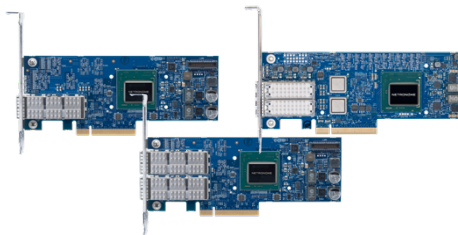


**Security Appliances** for Next Gen Firewall, Unified Threat Management, Intrusion Protection



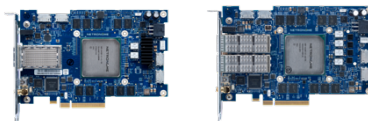
**NFP (Network Flow Processor)**  
NFP 4/6K Silicon based on Intel 22nm FinFET technology

NFP silicon family contains up to 120 processing cores and supports 10 to 100GbE speeds



**Agilio CX SmartNICs** with PCIe Gen3x8 and 2x10/25/40GbE network ports

Low profile PCIe adapters consume <25W, enable up to 50GbE throughput and saves as many as 12 x86 CPU cores



**Agilio LX SmartNICs** with PCIe 2xGen3x8 and 2x40GbE & 1x100GbE network ports

Full height dual PCIe adapters deliver up to 100GbE throughput and saves as many as 18 x86 CPU cores



**Agilio Software** supports Open vSwitch, Open Contrail and Linux Firewall acceleration

Turnkey server networking acceleration software packages deliver rich networking features with full VM mobility

Small kernel based virtual machine

10 64 bit registers

512 byte stack


Max 4k RISC bytecode instructions

Infinite size key-value stores (maps)

- ▶ Maximum of 64 maps per program

BPF has a strict verifier to ensure programs do not contain loops or other non-permitted state

Helper functions to do any essential work outside of BPF (e.g. map lookups)

A yellow rectangular box with a thin black border containing the text "4k eBPF Bytecode instructions".

4k eBPF  
Bytecode  
instructions

A stack of five blue rectangular boxes with thin black borders. The top box contains "r0", the second "r1", the third a dot, the fourth a dot, and the bottom box "r10".

r0


r1

•

•

•

r10

An orange rectangular box with a thin black border containing the text "512 byte stack".

512 byte  
stack

A teal rectangular box with a thin black border containing the text "Maps".

Maps



A group of fully programmable FPC

These are arranged into islands containing 12 FPCs each

Combined with specialist islands containing FPCs + hardware blocks

In total between 72-120 FPCs

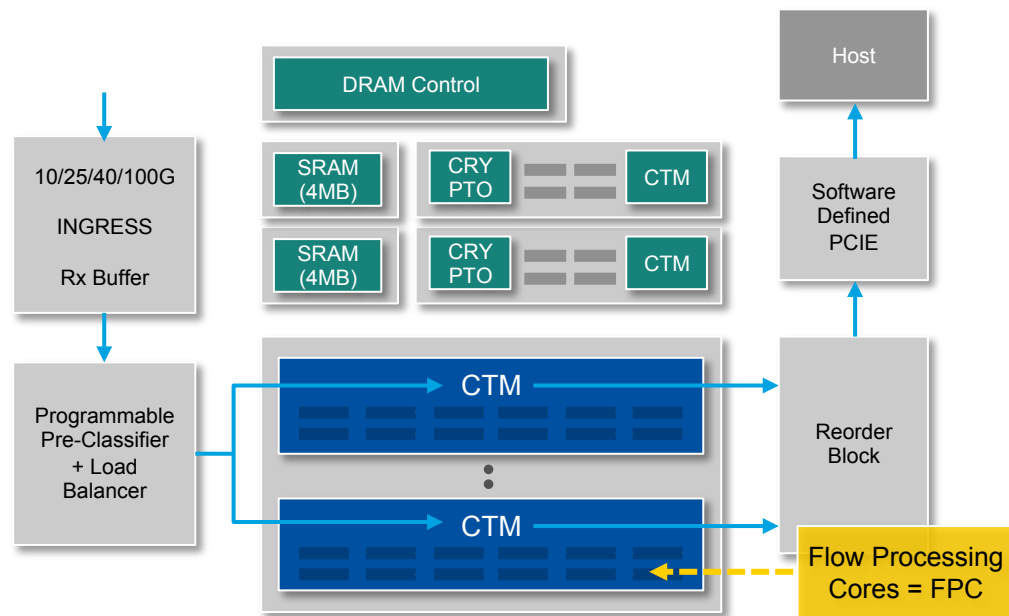
320KB SRAM memory per island

- ▶ 64 KB CLS-Cluster Local Scratch
- ▶ 256 KB CTM-Cluster Target Memory

8 MB of SRAM per NIC

2 GB of DRAM per NIC

RX Path



Each FPC has the ability to run 4 or 8 cooperatively multiplexed threads

128 32 bit A registers and 128 32 bit B registers

- ▶ A bank registers can only interact with B bank registers

256 32 bit transfer registers-128 read, 128 write

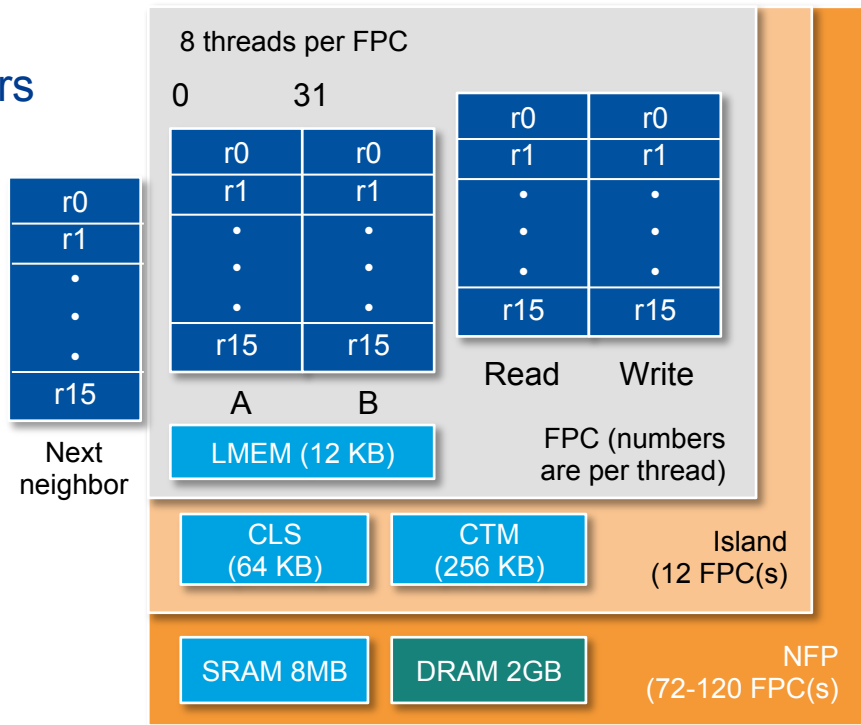
128 32 bit next neighbor registers

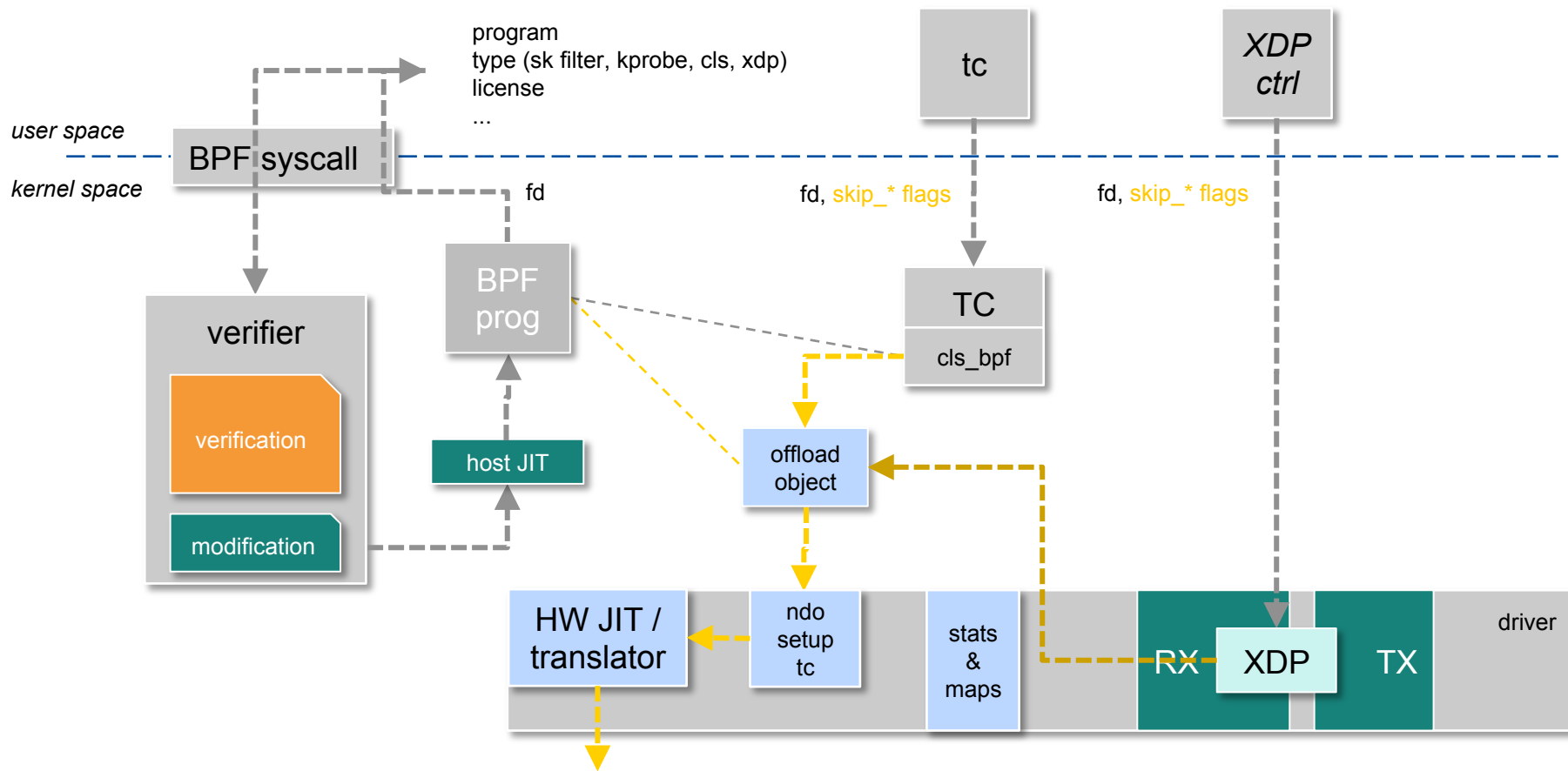
All of these registers are divided between the 4 or 8 contexts

- ▶ so 8 context leads to 16 A registers, 16 B registers etc. per thread

8k of instructions per FPC

12 KB of SRAM per FPC



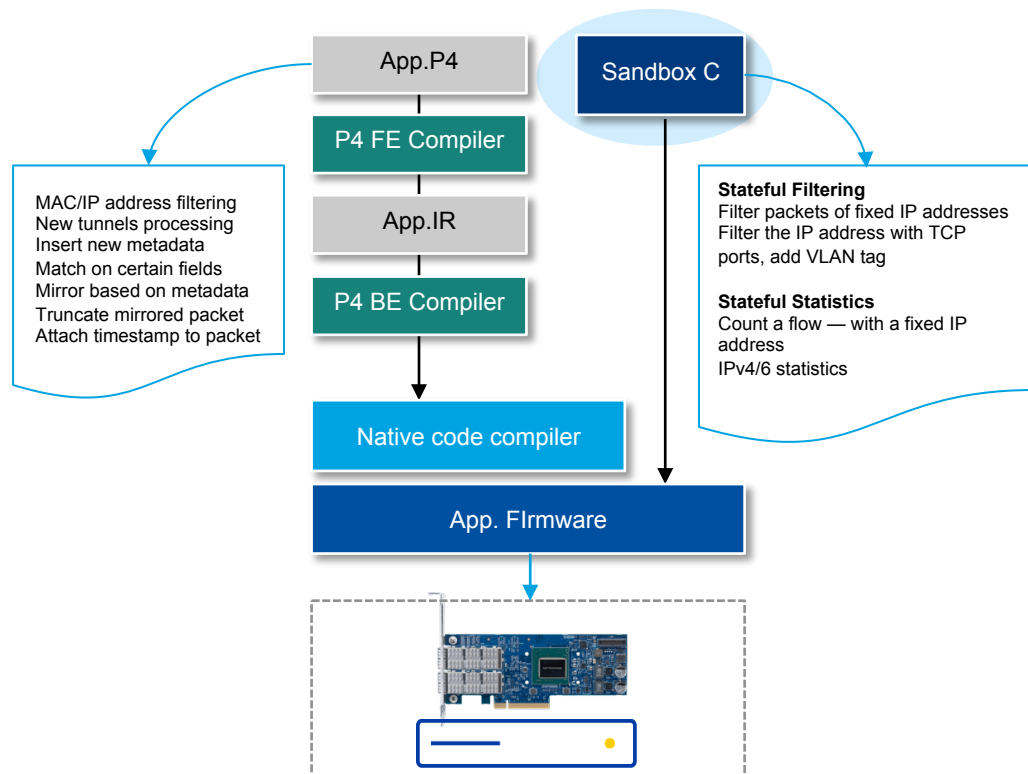




## Fully programmable hardware could do anything:

- ▶ If a consumer has the resources to be able to build custom code internally it may be preferred
- ▶ Will allow the consumer to gain higher performance than would otherwise be attainable
- ▶ The onus for ensuring maintainability and stability however then on the consumer

Should the community agree on a generalized architecture for querying SmartNICs and debugging?



Offload of applications allows the consumer to transparently offload processes that are already being used

Today an example of this would be OVS offload

Currently done using kernel hooks in OVS

Moving to a more generalized TC based architecture

Both Netronome and Mellanox are currently working on this upstream

