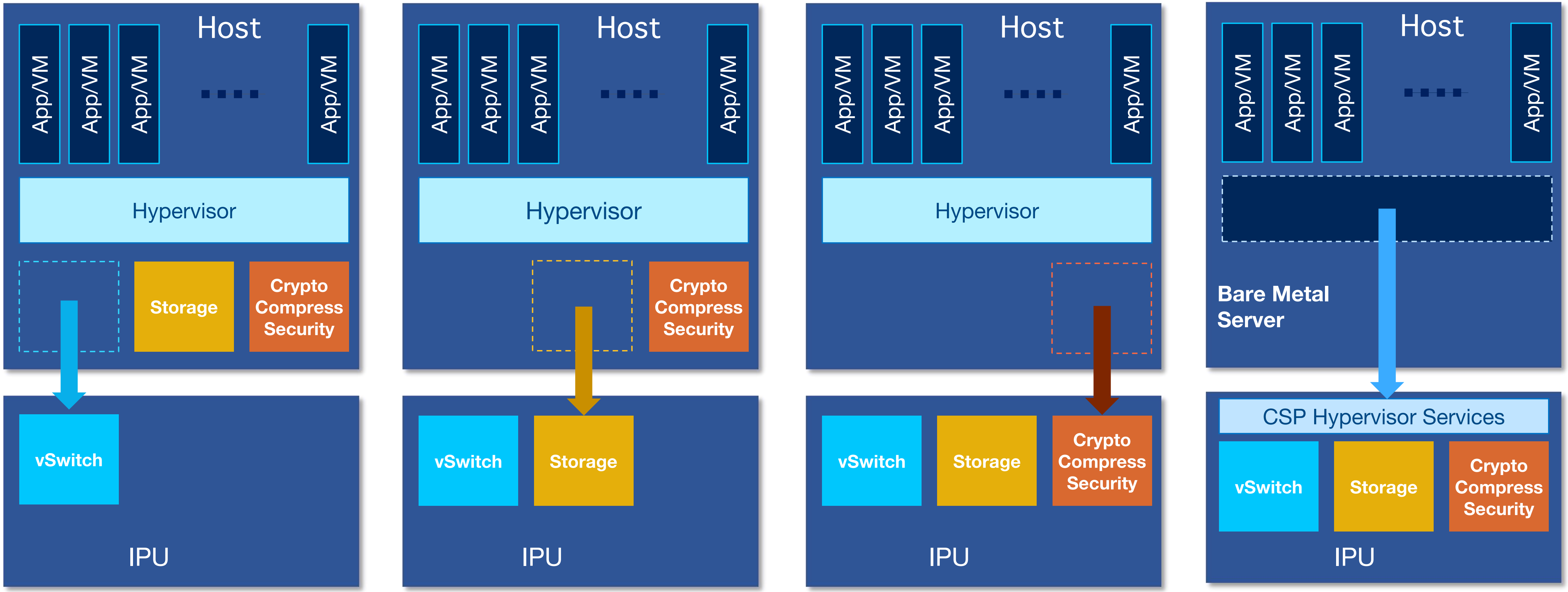
KUBERNETES COMMUNITY DAYS BEIJING 2023

KUBERNETES
COMMUNITY DAYS | BEIJING

Intel IPU在云数据中心中的实践与探索

臧锐

Leading CSPs are Driving Infrastructure Acceleration



1
- vSwitch widely deployed by CSPs
- First step for most customers
- Infrastructure services accelerated

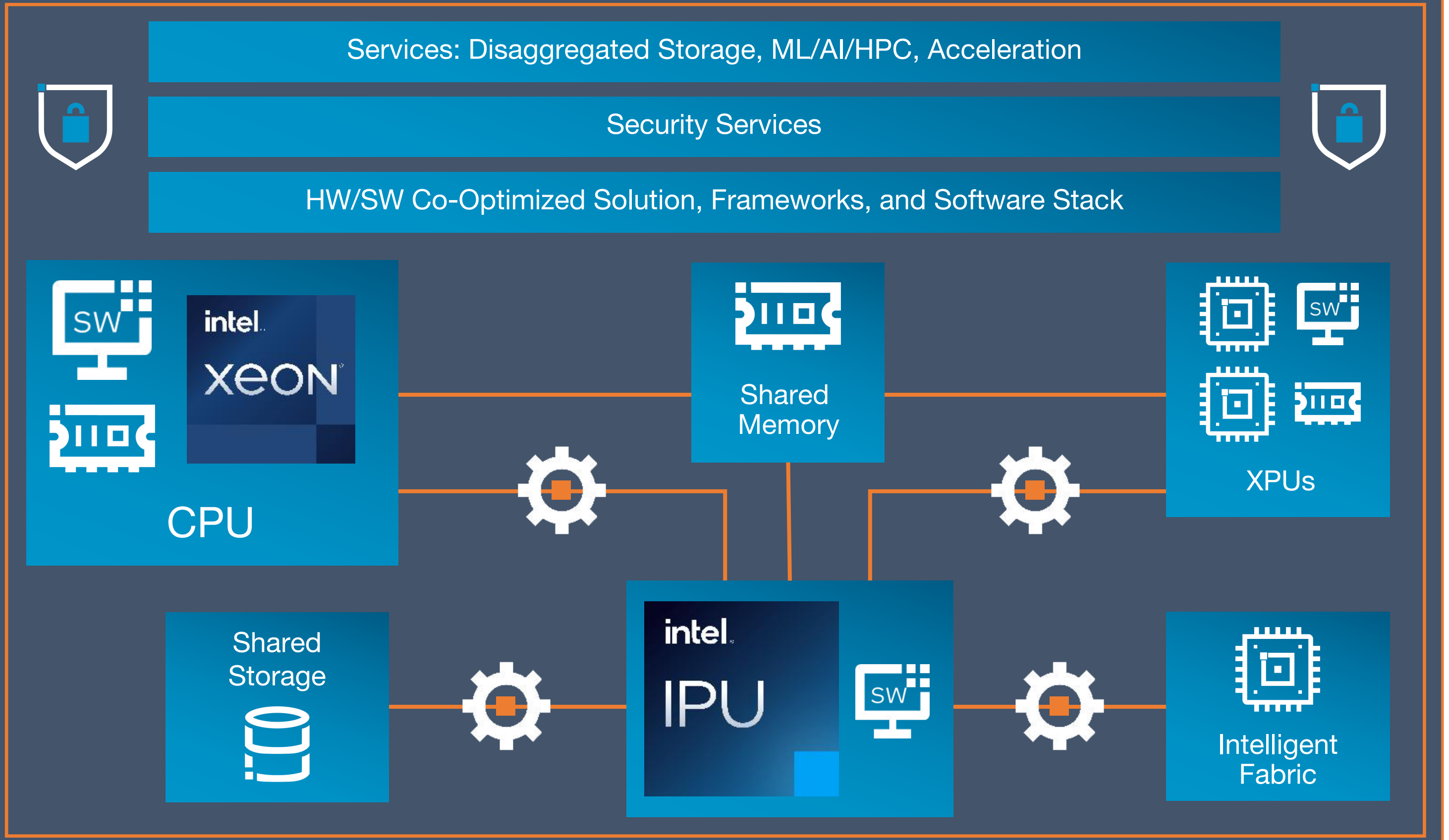
2
- Storage throughput increasing
- Second step for many customers

3
- Inline ops can be accelerated
- Often paired with storage

4
- Tenants rent entire *physical* server
- Bare metal value
  - Improved performance predictability and security for tenant
  - Better isolation for landlord

Intel IPU (Infrastructure Processing Unit)

Intel® IPUs are the Heart of a Disaggregated Heterogeneous Architecture with Co-optimized HW, Open-Source SW and Advanced Security & Manageability.



"General purpose processing will continue to play a critical role in the growth of warehouse scale computing, but we are increasingly seeing the need for a range of domain-specific accelerators for ML, data processing, and more. The IPU will play a central role in all of the above trends."

~ Amin Vahdat, Fellow and VP of Systems & Services Infrastructure, Google
Source: Intel Innovation event, October 27, 2021

Google Cloud

Intel IPU (Infrastructure Processing Unit)

开源生态

DPDK SPDK P4 ipdk Infrastructure Programmer Development Kit

开放的软件生态

通过同不断扩大的生态链领先厂商合作提供IPU解决方案

服务云、运营商、企业及其它行业客户

领先的产品组合

2021

FPGA+IA platform
Big Springs Canyon

2022

2nd Gen 200G

FPGA+IA platform
Oak Springs Canyon

ASIC
Mount Evans

当前

2023/2024

3rd Gen 400G

intel. IPU

FPGA+IA platform
Hot Springs Canyon

ASIC
Mount Morgan

未来发布

2025+

4th Gen 800G

intel. IPU

FPGA+IA platform
Next Gen

ASIC
Next Gen

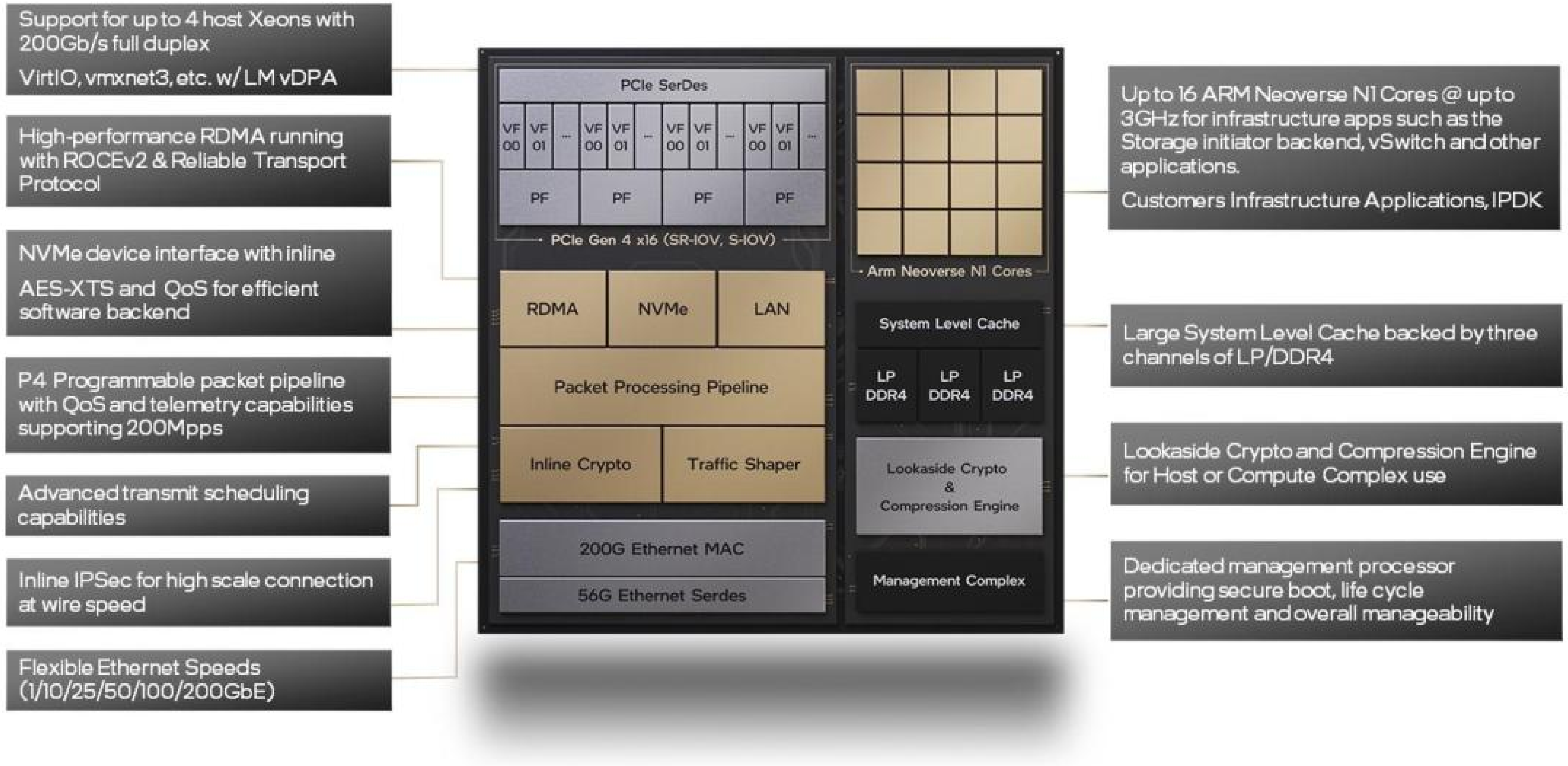
下一代

FPGA + Xeon-D IPU

	
Big Spring Canyon Intel FPGA IPU C5000X-PL	Oak Springs Canyon Intel FPGA IPU C6001X-PL
OVS，存储（NVMe-oF），安全	OVS，存储（NVMe-oF），安全
2x25G （50G带宽）	2x100G（200G 带宽）
针对云和运营商的存储、OVS、安全等方面的需求	针对云和运营商的存储、OVS、安全等方面的需求
需要部分定制开发	Intel OFS

在国内多家云服务商实现产品商用，加速功能涵盖网络、存储、安全等

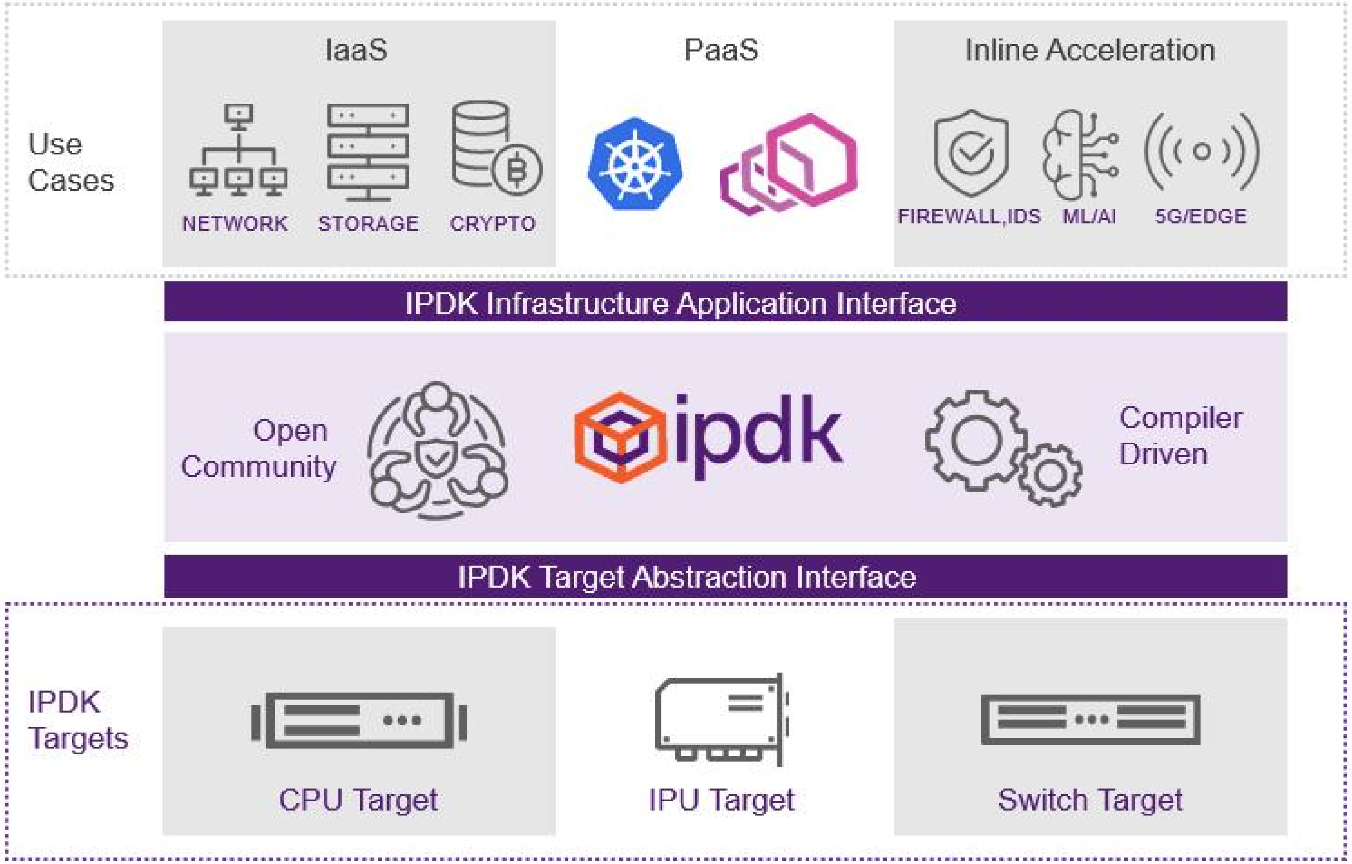
ASIC (Mount Evans) IPU



- ❑ 同国际云服务商联合开发，基于英特尔先进技术实现网络、存储、安全、加解密等加速功能
- ❑ 正在同国内主流云服务商合作进行产品试用和解决方案定制

IPDK – Infrastructure Programmer Development Kit

IPDK is a **community-driven, target agnostic** framework for **infrastructure programming** that runs on a CPU, IPU, DPU, or switch.



IPDK.io: Infrastructure Programmer Development Kit
Collaborate with the community on Github & Slack

IPDK.io : Infrastructure Programmer Development Kit
Collaborate with the community on Github & Slack

Slide courtesy of IPDK

IPDK – Infrastructure Programmer Development Kit

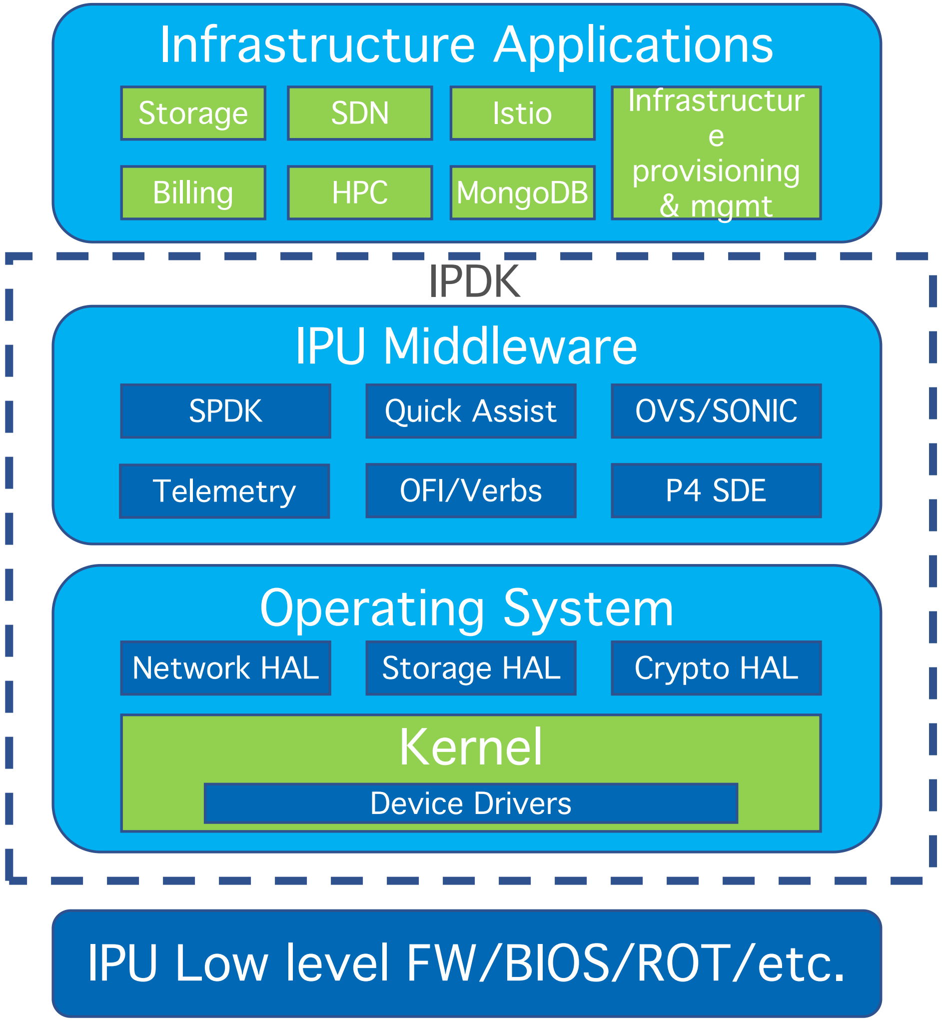
IPU SW stack includes:

1. Application
2. IPDK
3. low level SW/FW

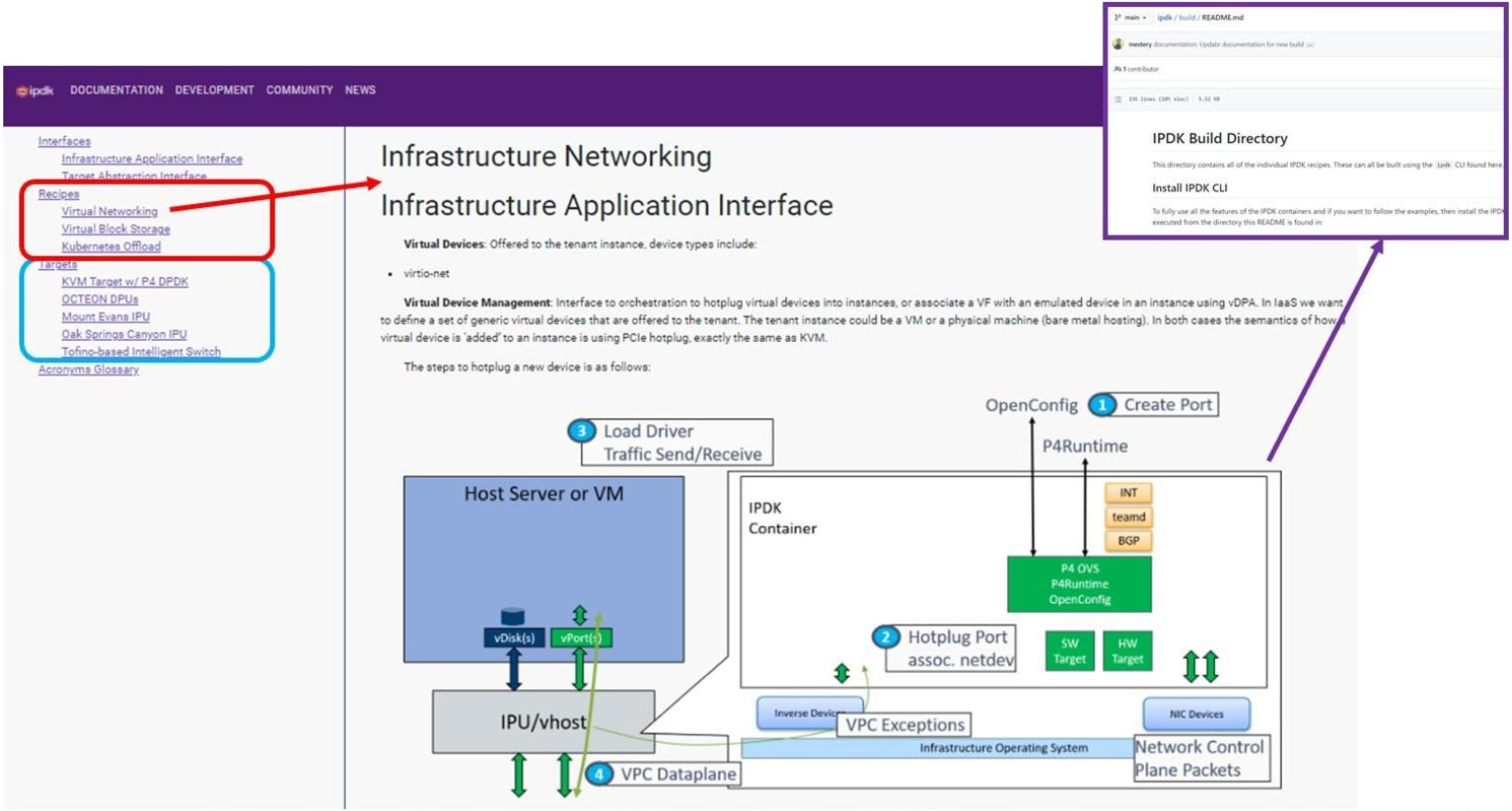
CSP & Open Source Infrastructure Applications
(examples shown)

IPDK is the open source abstraction for programming IPUs

Low level platform dependent SW

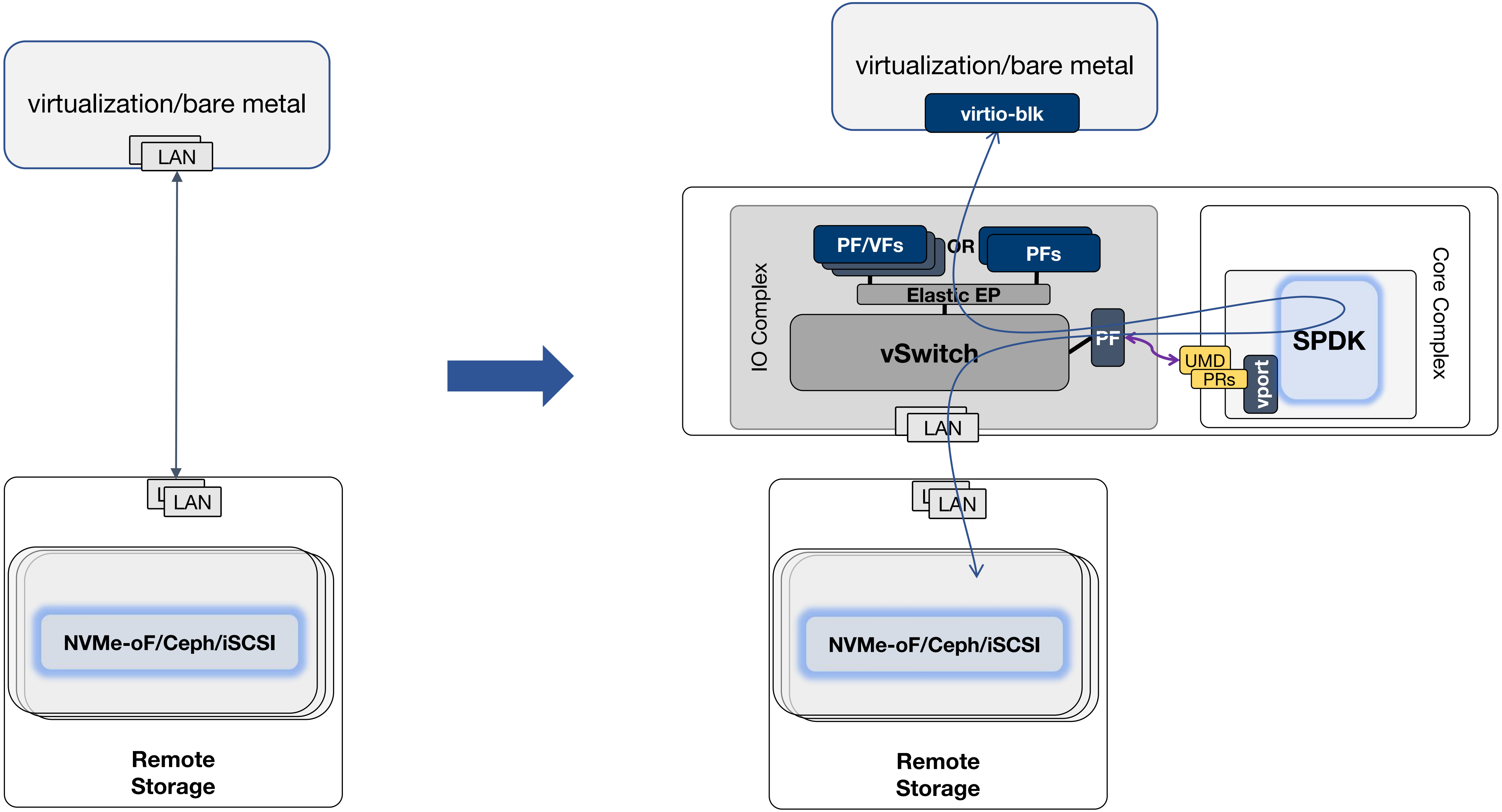


Use Case in IPDK



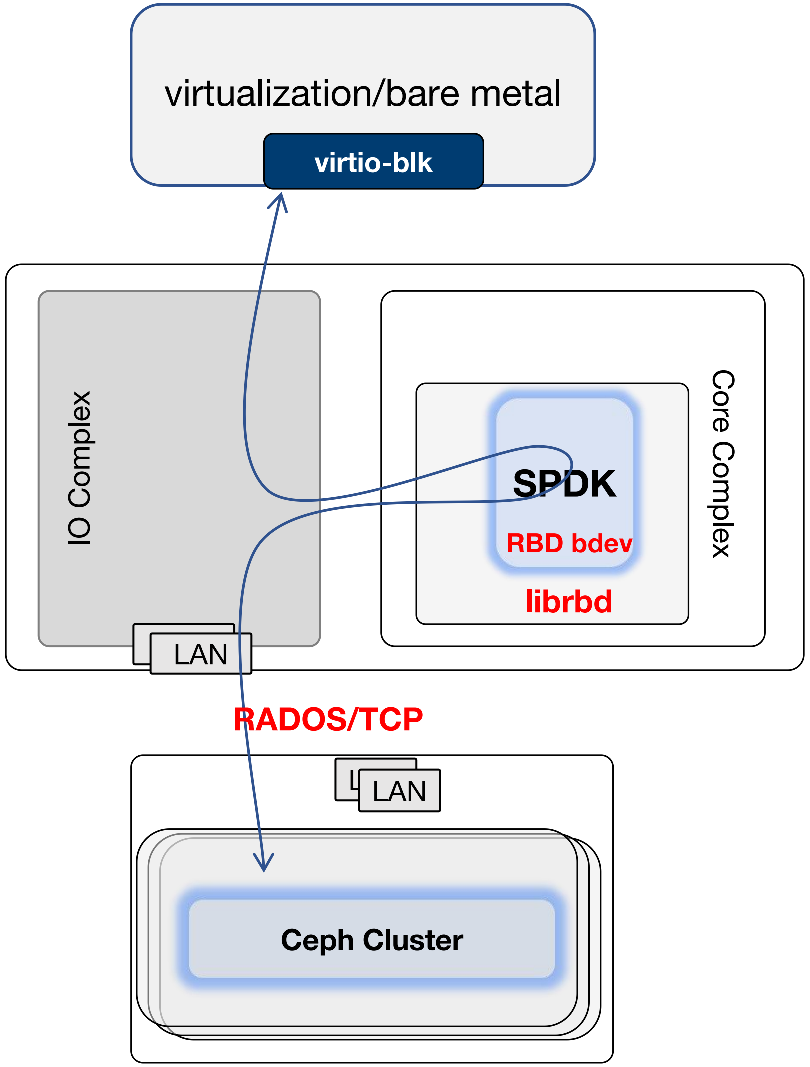
Distributed Scale-out IPU Storage

IPU Remote Storage

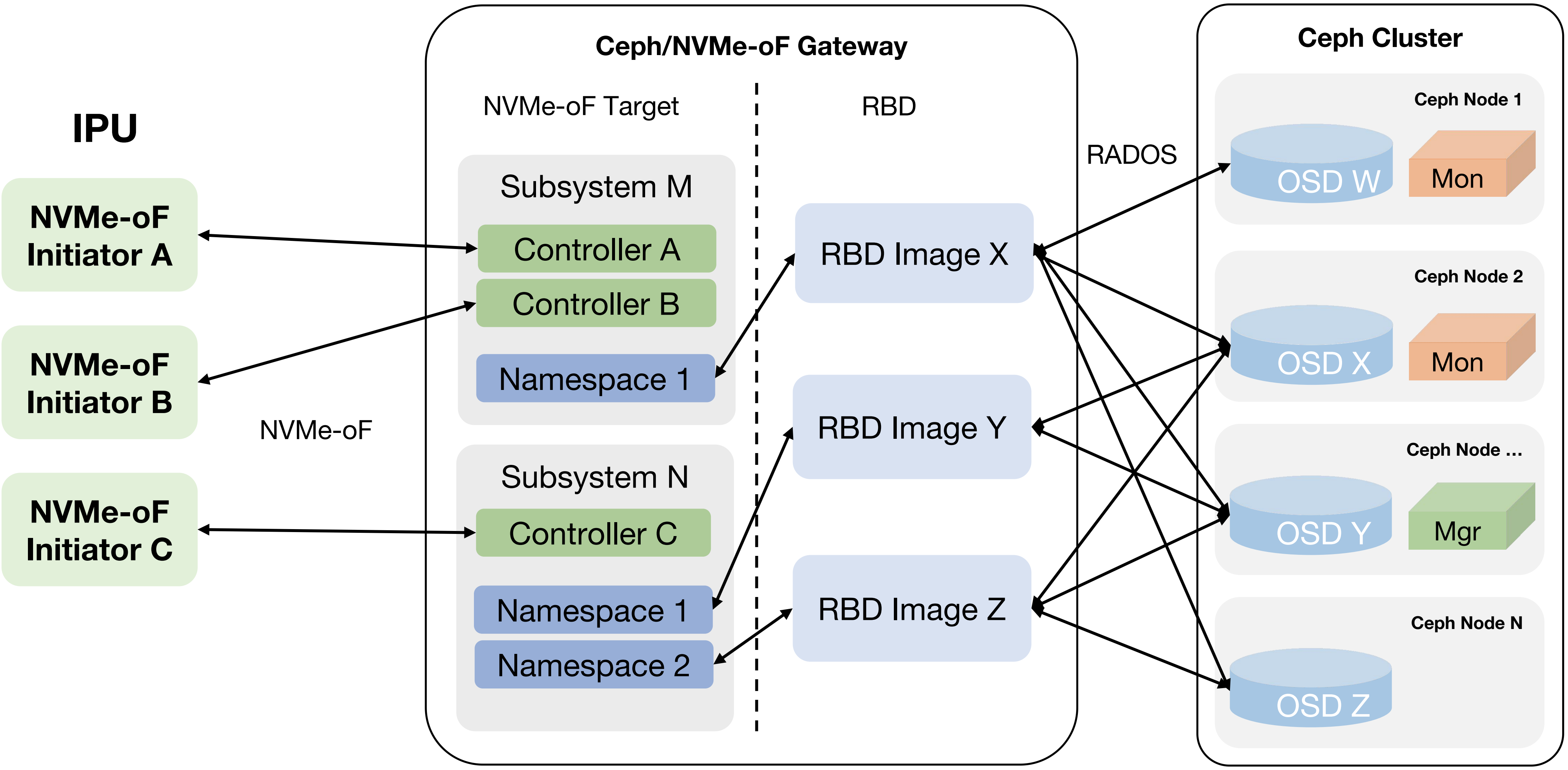


IPU Remote Storage based on Ceph

- Faster arm cores or Xeon–D cores for IPU
- Optimized librbd
- Protocol offloading, e.g. TCP, RADOS



Gateway: deployments of RBD over NVMe-oF



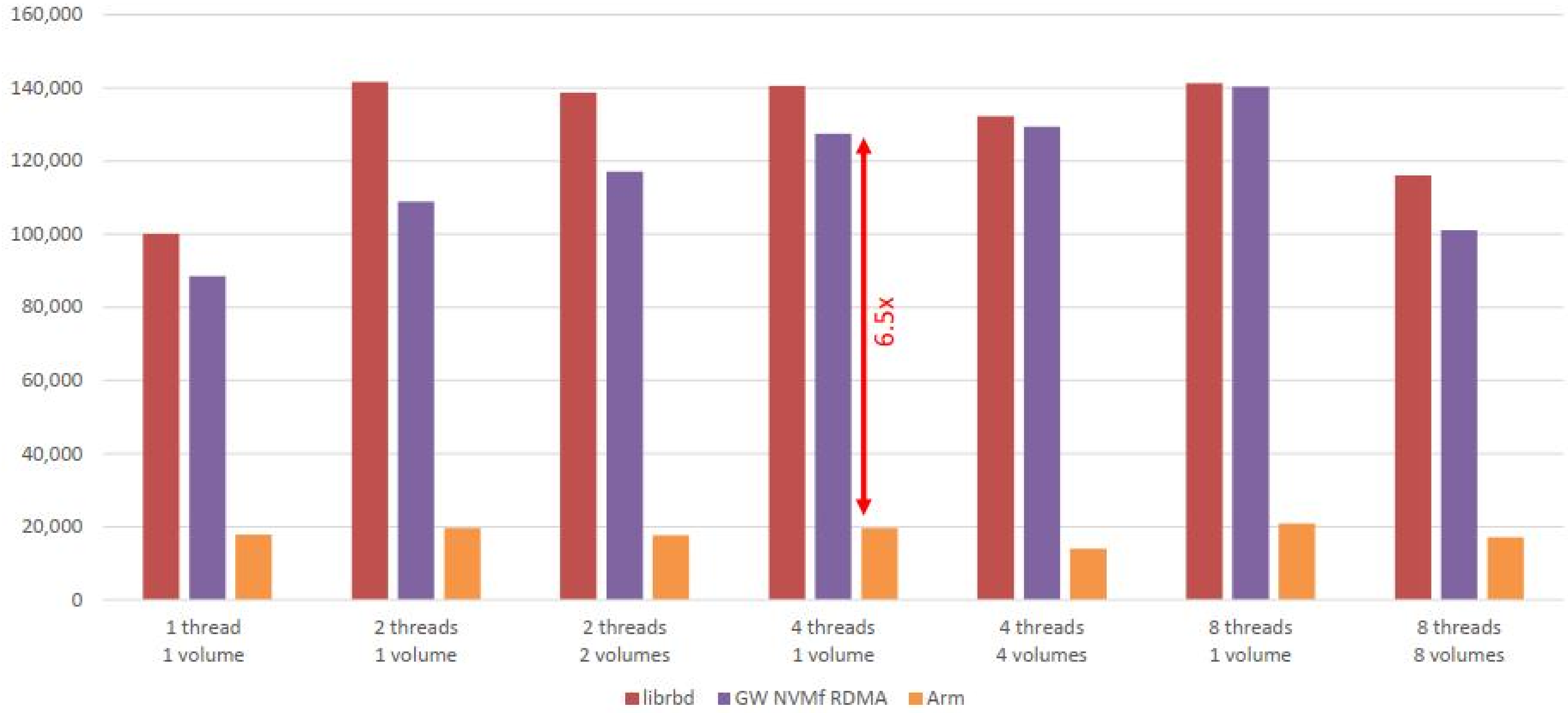
Gateway: deployments of RBD over NVMe-oF

- uses SPDK based gateways to connect to Ceph cluster to support the block device operations

  *ceph gateway: https://github.com/ceph/ceph-nvmeof

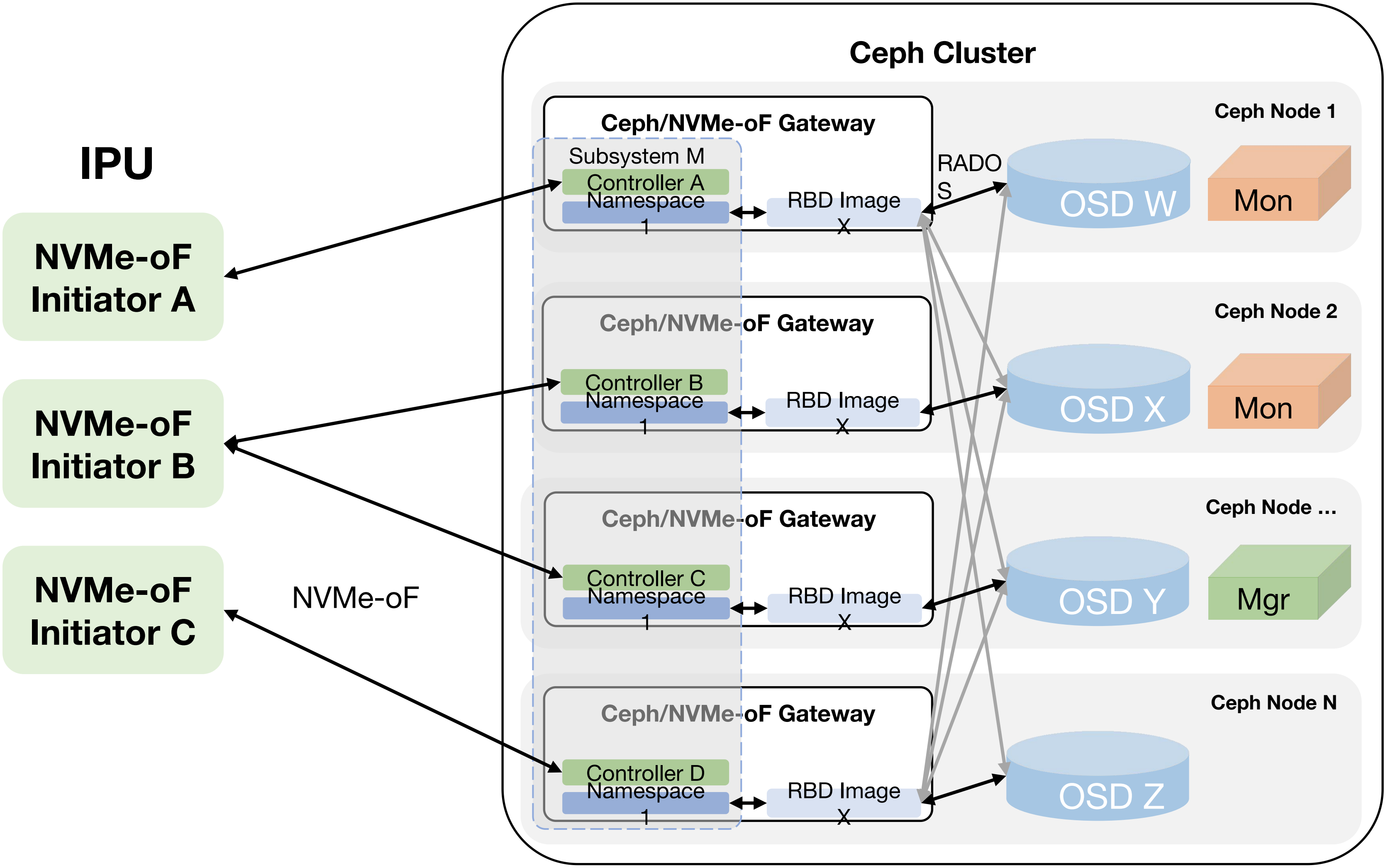
- be feasible but at the cost of extra compute resources and additional network hops
- lightweight client in IPU initiator, and host/IPU overhead much less than librbd

READ IOPS QD128@16KIB



* From Jonas Pfefferle (IBM)

Scale-out IPU Storage



Scale-out IPU Storage

- hosts/IPU route each NVMe IO to the correct node with hint
- NVMe-oF for public network in datacenter, and RADOS/TCP for internal network in remote storage backend
- eliminate dedicated gateways and extra hop
- lightweight client in IPU initiator, and host/IPU overhead much less than librbd (more offload friendly)
- can be easily extended to support various storage backend besides Ceph

Accelerating Faas/Container Image Construction via IPU

FaaS/Container image offloading motivation

- FaaS are usually the short programs (functions), it’s critical to execute such a function in a fast way without any unnecessary overhead of preparations.
- The main overhead comes from compiling the code into executable binary, pack the executable binary with required libraries into a file system, and then get the container environment up running.
- For FaaS deployed in containers, the following are the main places for optimizations to reduce the overhead of preparations:
  1. Quickly build the FaaS/Container related images.
  2. Get container execution environment ready asap, including unpacking the image of FaaS into a file system.
  3. Expose the bundle to the container, then execute the FaaS applications in the selected container.

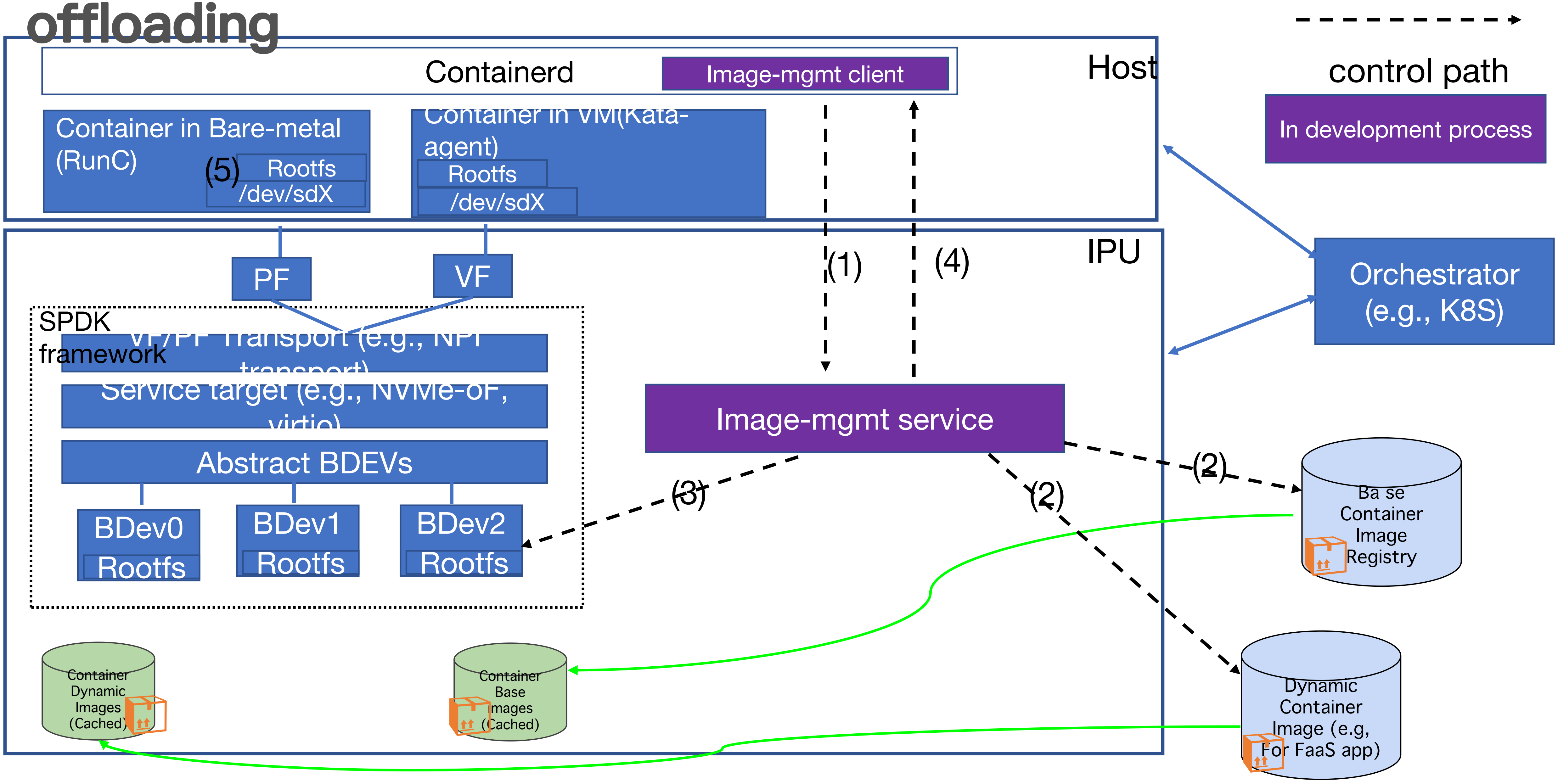
IPU can help the container bundle offloading

- As noted earlier, one of the main overhead of container comes from the preparation work, including:
  - Image pulling & file system bundle (e.g., rootfs) preparation.
  - Start runtime shim.
  - The selected runtime class started to run (e.g., RUNC).
- IPU can be used to accelerate container image pulling & file system bundle preparation, proposed by “Ziye Yang, Yadong Li and Jun Zeng” in SDC 2022
  - The container image related operations can be moved into IPU.
  - IPU accelerators can be used for image decompression, decryption, etc.
  - IPU can cache the images and enable sharing of the unpacked image layers.

Ingredients for the DPU/IPU Solution

Ingredient	IPU Support	Benefit from IPU based solution
Block Device Interface to Host	NVMe, virtio-blk interfaces	Fully leverage on IPU based storage disaggregation solution
Block device Hot-plug	Yes, IPU designed for bare-metal and virtualization usages	Align with IPU/DPU’s long term strategy as a control point in data center
Container image pulling & caching	Download container images from image registry and cache images	IPU as a control point is the idea choice to manage container images
Container Rootfs construction	Construct the rootfs in an assigned bdev provided by the block service target in the IPU	IPU can offload such work from the host
Unpacking rootfs or sharing filesystem among containers	Leverage the snapshot or cloning features of the bdev in block service target are required.	IPU can offload such work efficiently because of integrated accelerators such as decompress engines.
Control path communication between IPU and container runtime software	Provide related RPC service to interact with container management software.	Such RPC service is supported in IPU’s architecture and design

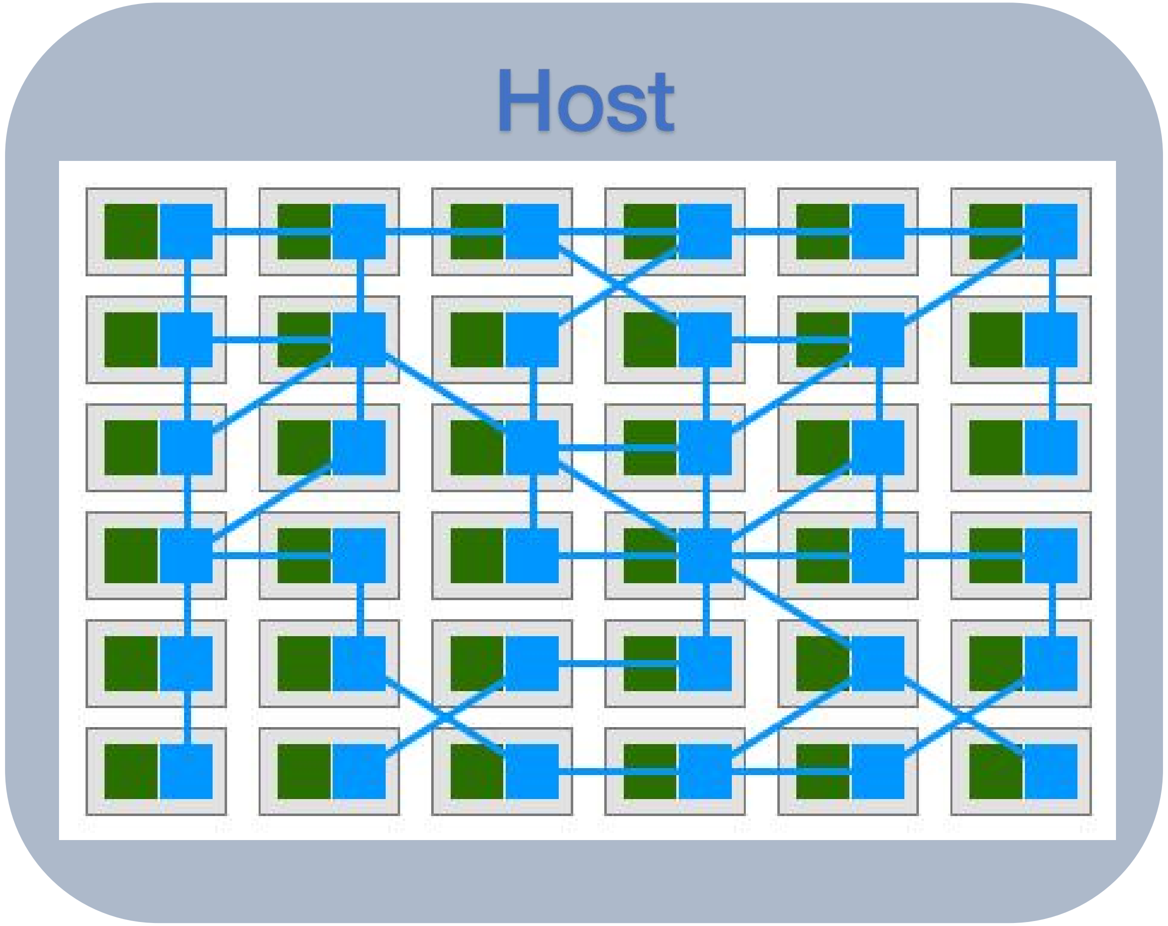
Key software components and flows by DPU/IPU offloading



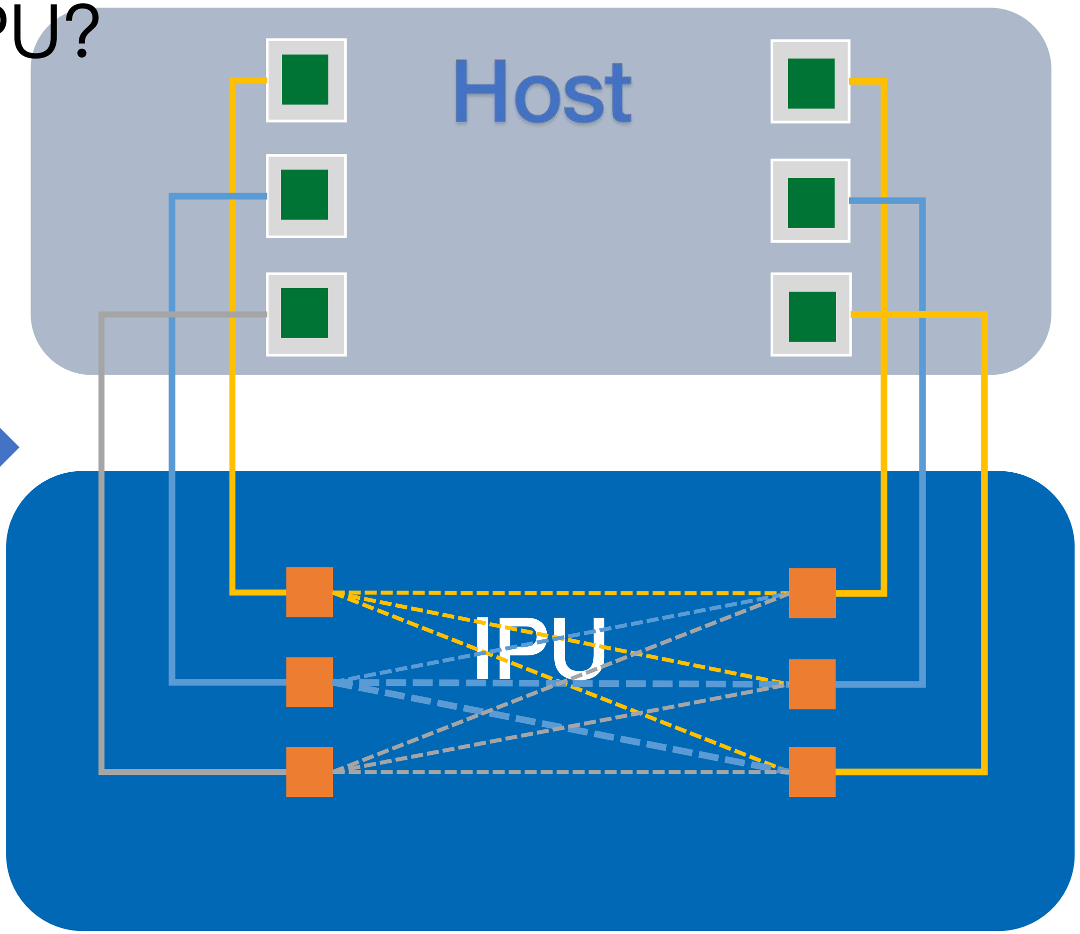
service mesh offloading with IPU

Imagination about service mesh

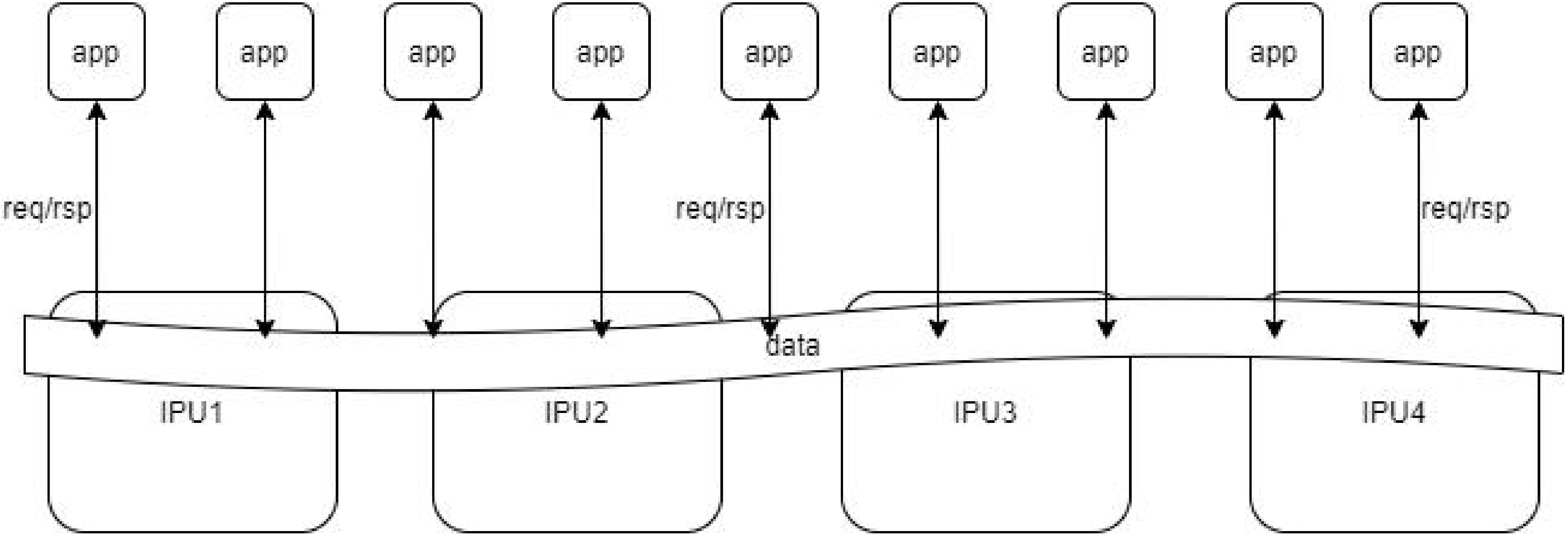
We are now in sidecar mode



Can we provision the mesh in IPU?



In another perspective

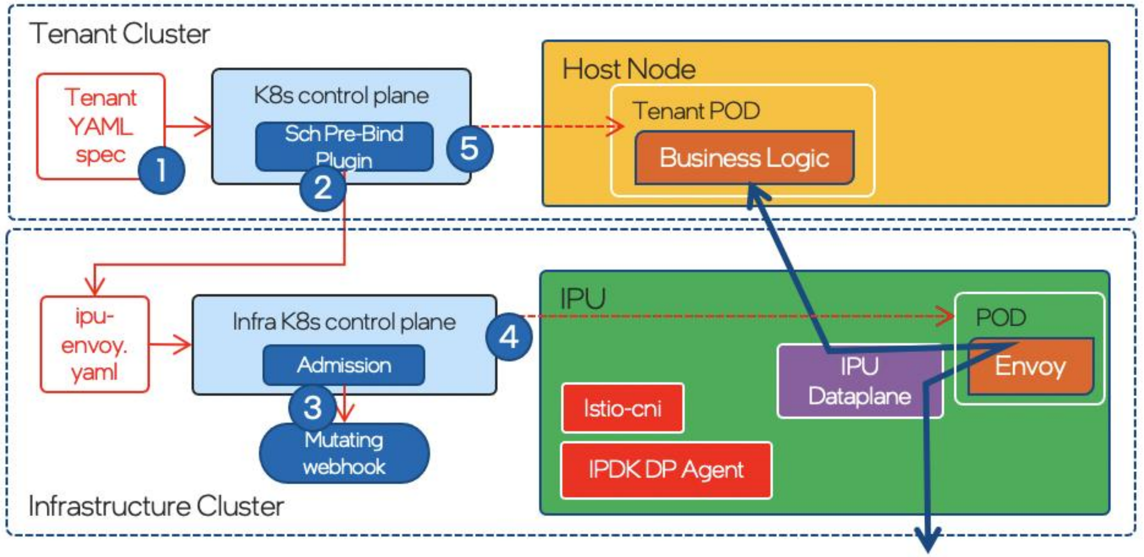


All the data are flowing between the IPUs, applications only take care of its own request/response

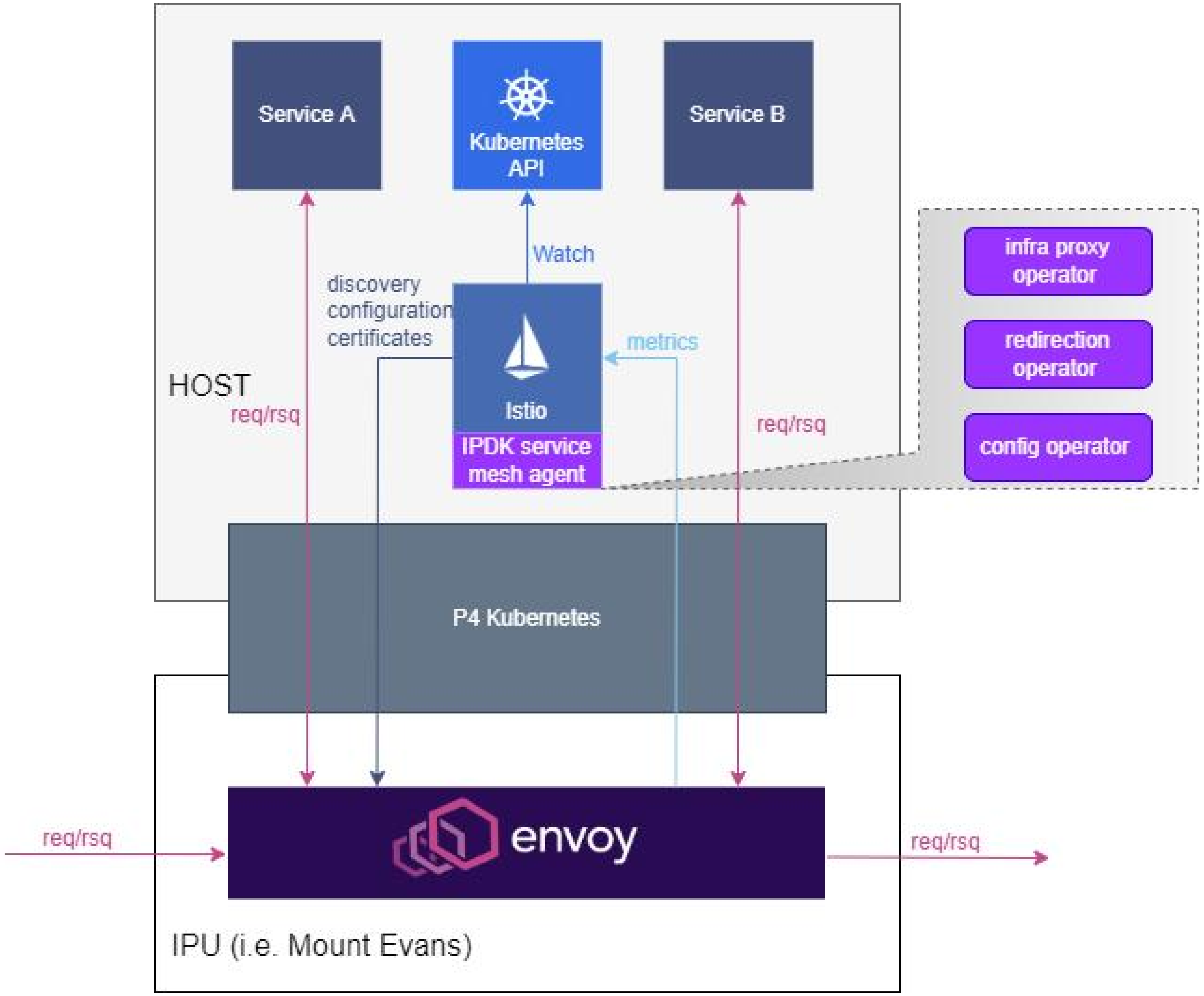
offloading ideas

envoy running on IPU's ACC w/o or small code changes

Split sidecar onto IPU
(Separation between Tenant & Infrastructure)



Description of the functionality



The IPDK service mesh agent enables users deploy envoy onto IPU as an infrastructure proxy instead of the "sidecar" along with every application. The subcomponents are:

- Infra proxy operator reacts to the service mesh namespace's label change and creates or destroy an instance of envoy running as infrastructure proxy on IPU.
- Redirection operator configure the traffic flows to P4 Kubernetes to redirect the traffic from applications to infrastructure proxies
- Config operator helps to send the DNS, external service entries' config to infrastructure proxies.

The Envoy is deployed as infrastructure proxy running on IPU's SoC, the service applications and infrastructure proxy are connected by the IPDK data plane (P4 Kubernetes) with accelerated data paths.

P4 Kubernetes is running across host and IPU to provide the data plane to the mesh, connect the service applications and infrastructure proxies through IPU's hardware pipeline.

Description of the functionality

This project depends on below external components.

1. Kubernetes(including API server and other Kubernetes components)
   Service mesh control plane runs on Kubernetes, user can add applications deployed in that cluster to service mesh, extend the mesh to other clusters.
2. Istio
   The Istio running as service mesh control plane takes user's desired configuration from Kubernetes, and its view of the services, and dynamically programs proxy nodes, updating them as the rules or other environment changes.
3. Envoy
   The envoy running as a set of intelligent proxies of service mesh. These proxies mediate and control all network communication between microservices. They also collect and report telemetry on all mesh traffic.
4. P4 Kubernetes
   The P4 Kubernetes project publishes open-source CNI p4 data-plane plugin components that would help offload the Networking rules from Calico CNI (Container Network Interface) to Intel IPU (MEV) platform. IPU customers can then use this open-source software in the GitHub repository to deploy their orchestration software.

Thanks.

KUBERNETES
COMMUNITY DAYS | BEIJING