


KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

Userspace Block Services based on UBLK and VDUSE in SPDK

Liu Xiaodong & Liu Changpeng
Intel

Agenda

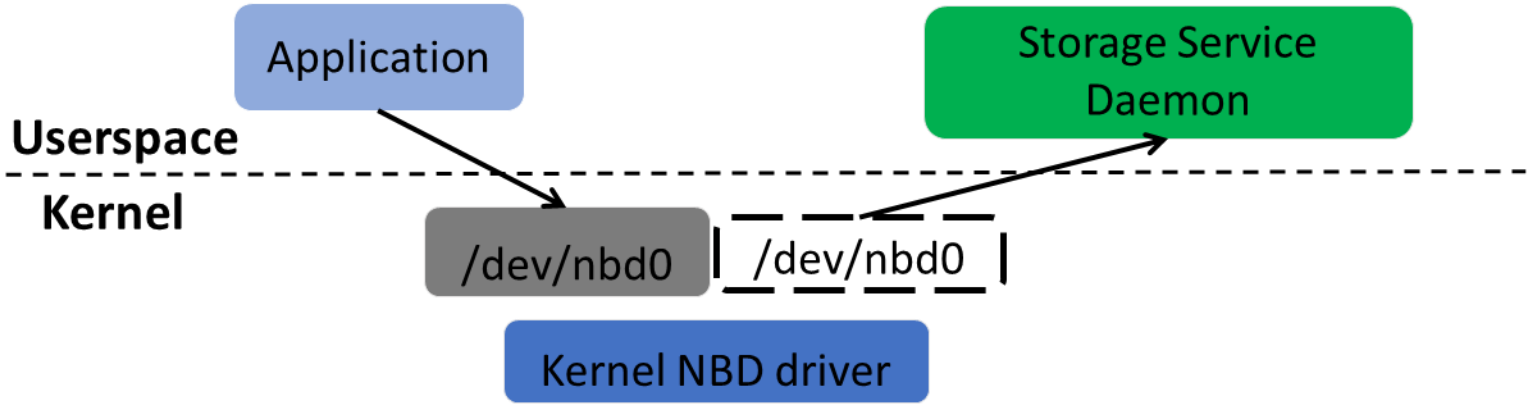
- Background
- Modern Design Comparing with Traditions
- VDUSE
- UBLK
- SPDK Progress
- Comparisons

Background

- Cloud Native / Container technology has already emerged as one of the hottest topics in the industry
- Kinds of APPs are shifted or shifting toward cloud native architectures
- Storage is an essential part for container as ephemeral or persistent data requirement
- SPDK, as widely adopted open source userspace storage solution, need to add container solutions support

Background

- Block storage for container is primarily provided by kernel directly
- Userspace daemon occasionally serves localhost IO requests based on specific protocols, like Linux NBD, iSCSI, etc.



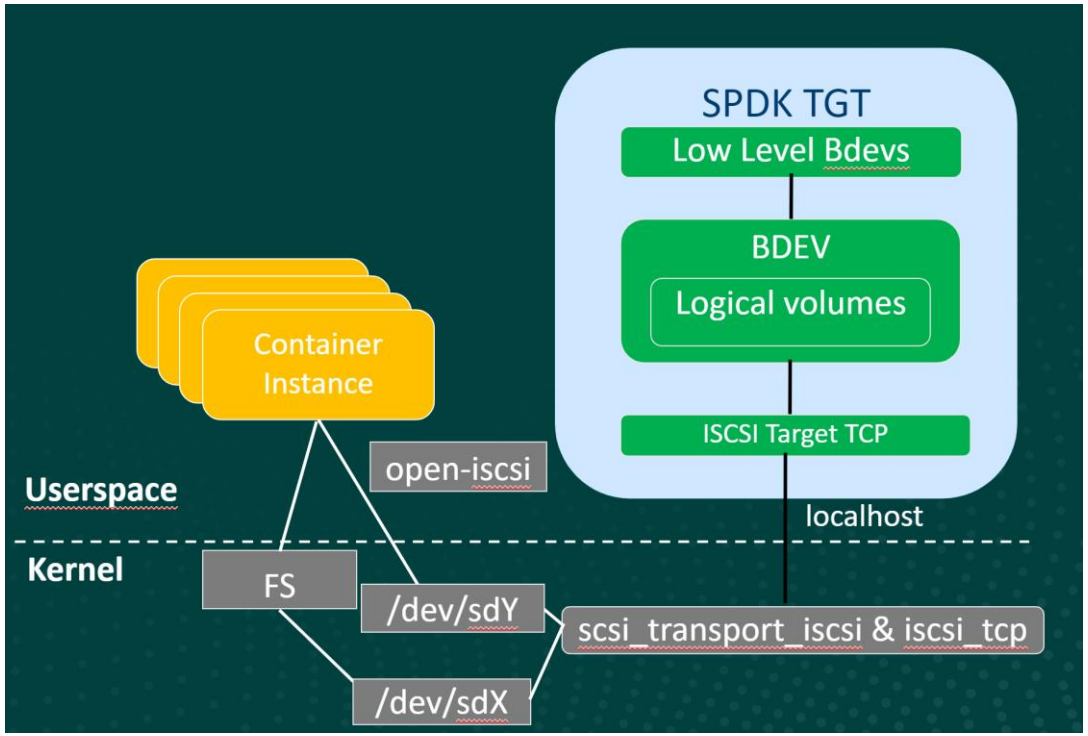
- Moving all block IO processing logics in userspace is more suitable to modern system environment
- We clarify the term “userspace block service” here refers to block service provided by userspace daemon to localhost via kernel.

Modern Design Comparing with Traditions

- iSCSI Target enables SCSI commands transfer between host and controller devices via standard IP network
- **TCP loopback mode** enables a circuitous path to present SCSI block device from userspace block service to localhost

SPDK iSCSI Target

- Multi-Queue isn't supported in initiator
- No core scaling based on session connection
- Not efficient

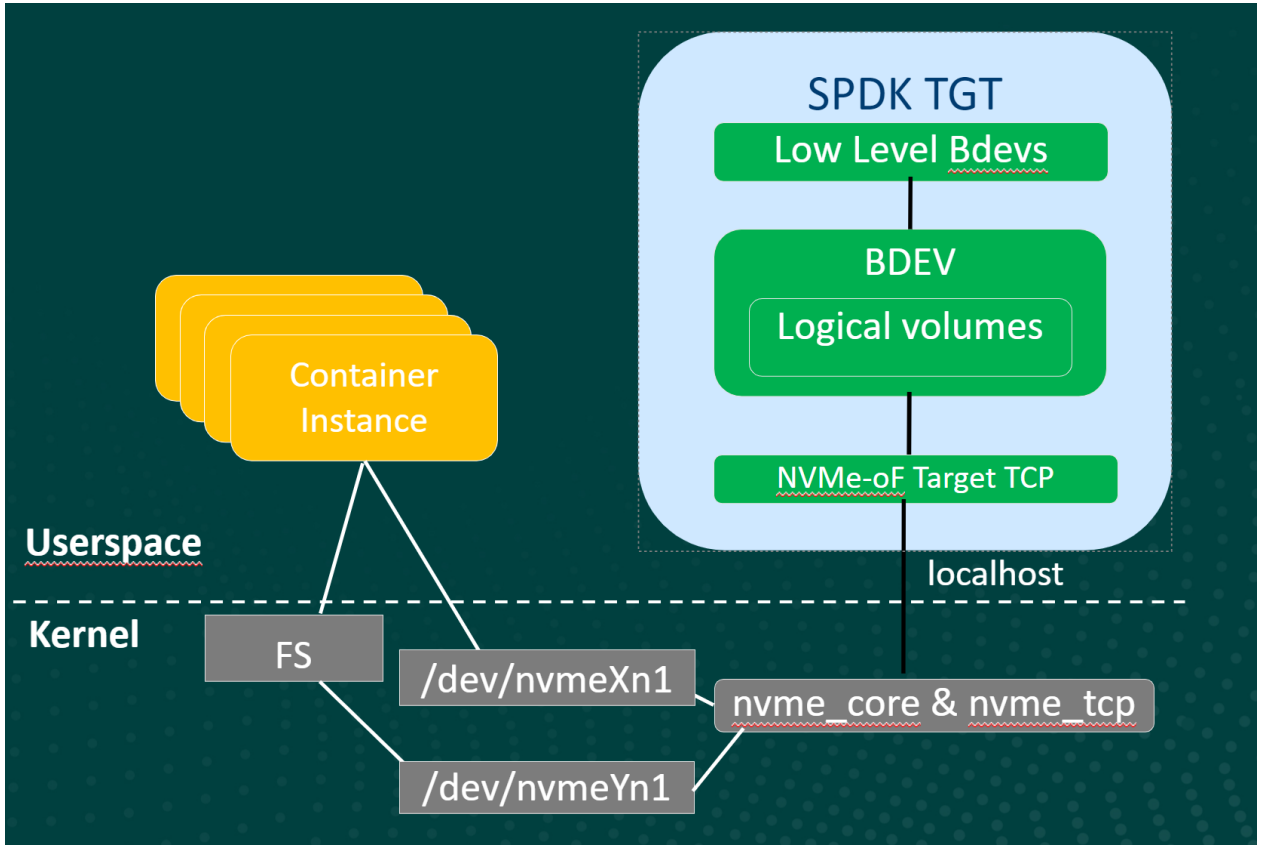


Modern Design Comparing with Traditions

- NVMe over TCP enables E2E NVMe operations between host and controller devices via standard IP network
- **TCP loopback mode** enables a circuitous path to present NVMe block device from userspace to localhost

SPDK NVMe-oF Target

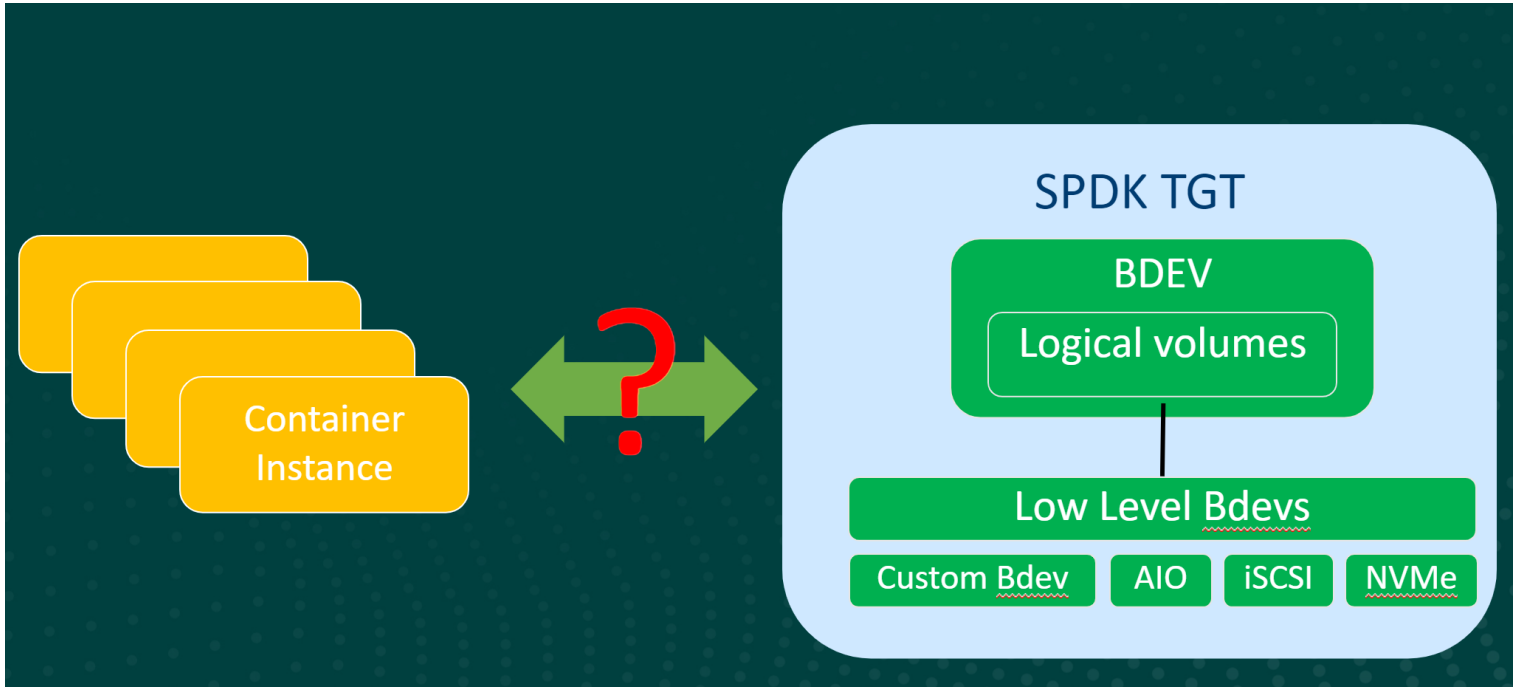
- Current recommended option
- Core scaling
- Well optimized
- Not efficient



Modern Design Comparing with Traditions

- Disaggregated Storage
  - NVMe-oF target, iSCSI target
- Virtualization Storage
  - vhost/vfio-user target
- Storage Libraries for App Integration
  - Bdev/Blobstore/Blobfs

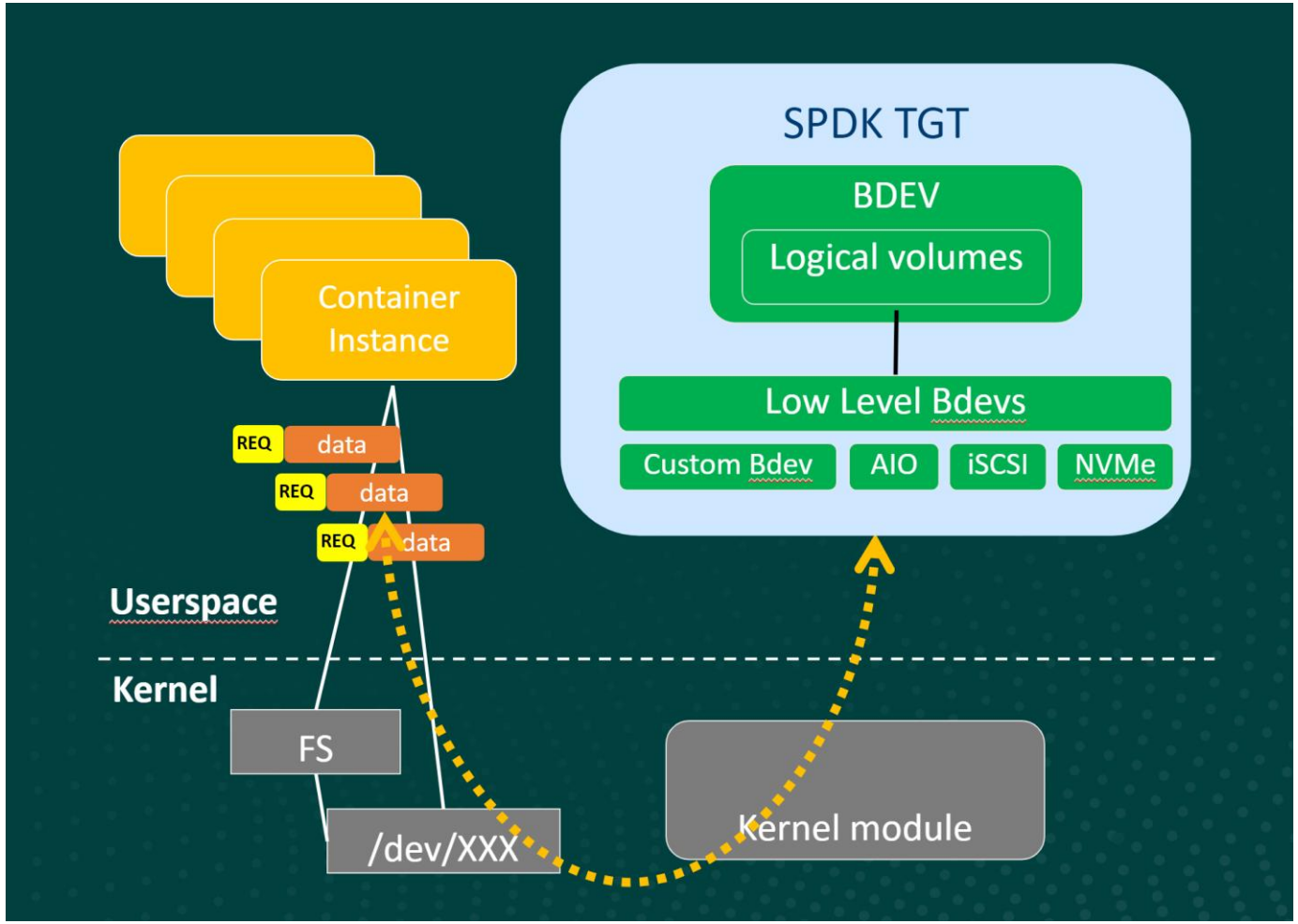
- How to support general containers efficiently?



Modern Design Comparing with Traditions

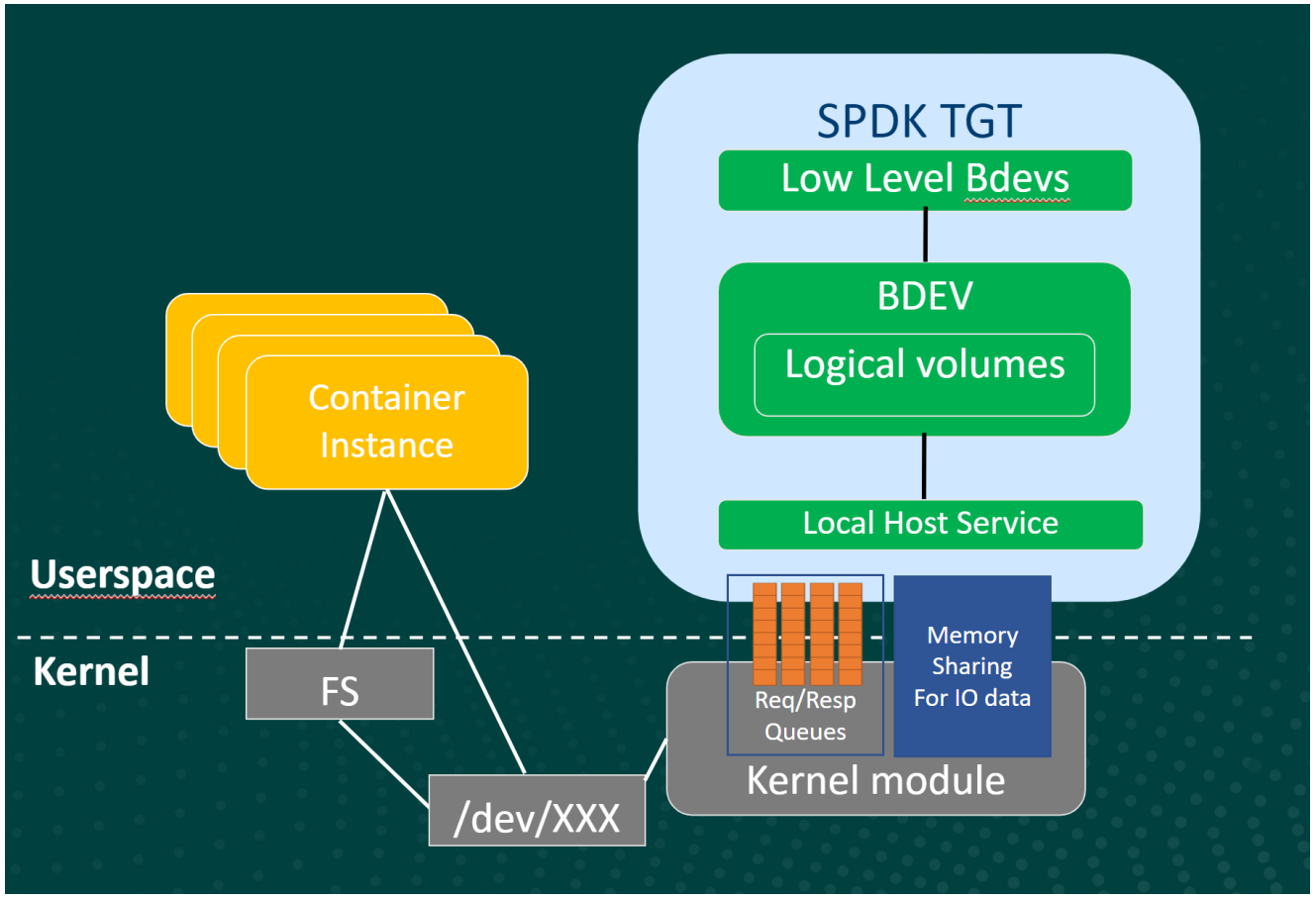
Key Requirements

- Notification interactions on IO request and IO completion
- Data transfer with less data copy
- Modern system requirement, like hot-upgrade, live-recovery



Modern Design Comparing with Traditions

- Principles to improve performance for userspace block service:
  - Queue based IO Req/Resp interacting
  - Memory shared IO data transferring
  - Efficient command pack/unpack operation
  - Service unavailable tolerance and restore



VDUSE

- Short for vDPA device in userspace
- Provides a way to implement vdpa devices in userspace
- Generic Virtio data path
- Enables workloads to reuse existing drivers

- Expose as **virtio-blk** device to localhost
- Requests are exchanged by **virtqueue**
- Data are copied once by software IOTLB from page-cache to **bounce buffer** mmapped in VDUSE backend.
- VDUSE, merged to linux kernel 5.5 in 2021 by Bytedance

VDUSE

- UMEM Registration feature in VDUSE:

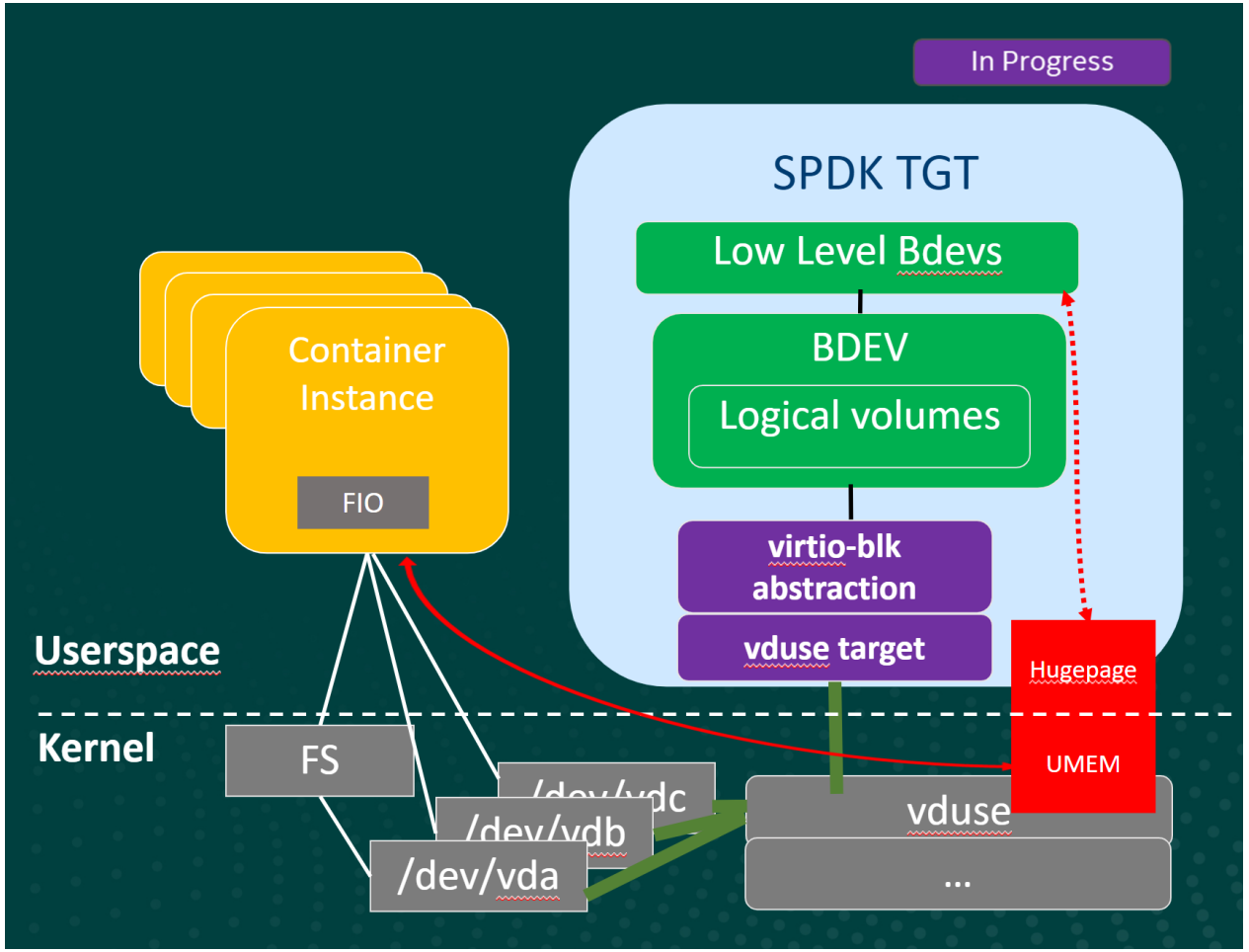
Originally, VDUSE backend can only mmap bounce buffer from kernel.

Now, VDUSE backend can register its own memory region as bounce buffer to kernel.

- Userspace driver accessible memory is required by NVMe device and RDMA in SPDK
- UMEM registration help SPDK construct efficient data path for a whole storage stack

VDUSE

- VDUSE backend in SPDK runs in a similar way as vhost target
  - Exchange IO requests via virtqueue
  - Access IO data via shared memory
  - Receive kick notification via eventfd
- But
  - Initialize, query and update device configuration control actions are via IOCTLs
  - Inject IRQ via IOCTL
- Similar performance expectation with SPDK vhost-blk per reactor



UBLK

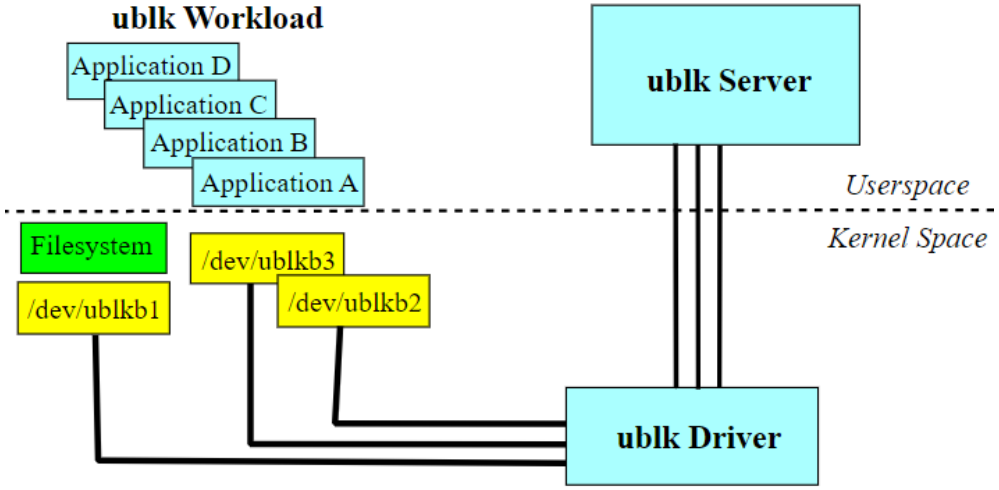
UBLK: io_uring based generic Userspace Block Device

- Target for cloud native storage and distributed block storage's host block device interface
- High performance
- Expose as generic block device to host with multi-queue support
- Block IO processing in userspace block service
- UBLK, merged to linux kernel 6.0 in 2022, io command via io_uring pt cmd.

UBLK

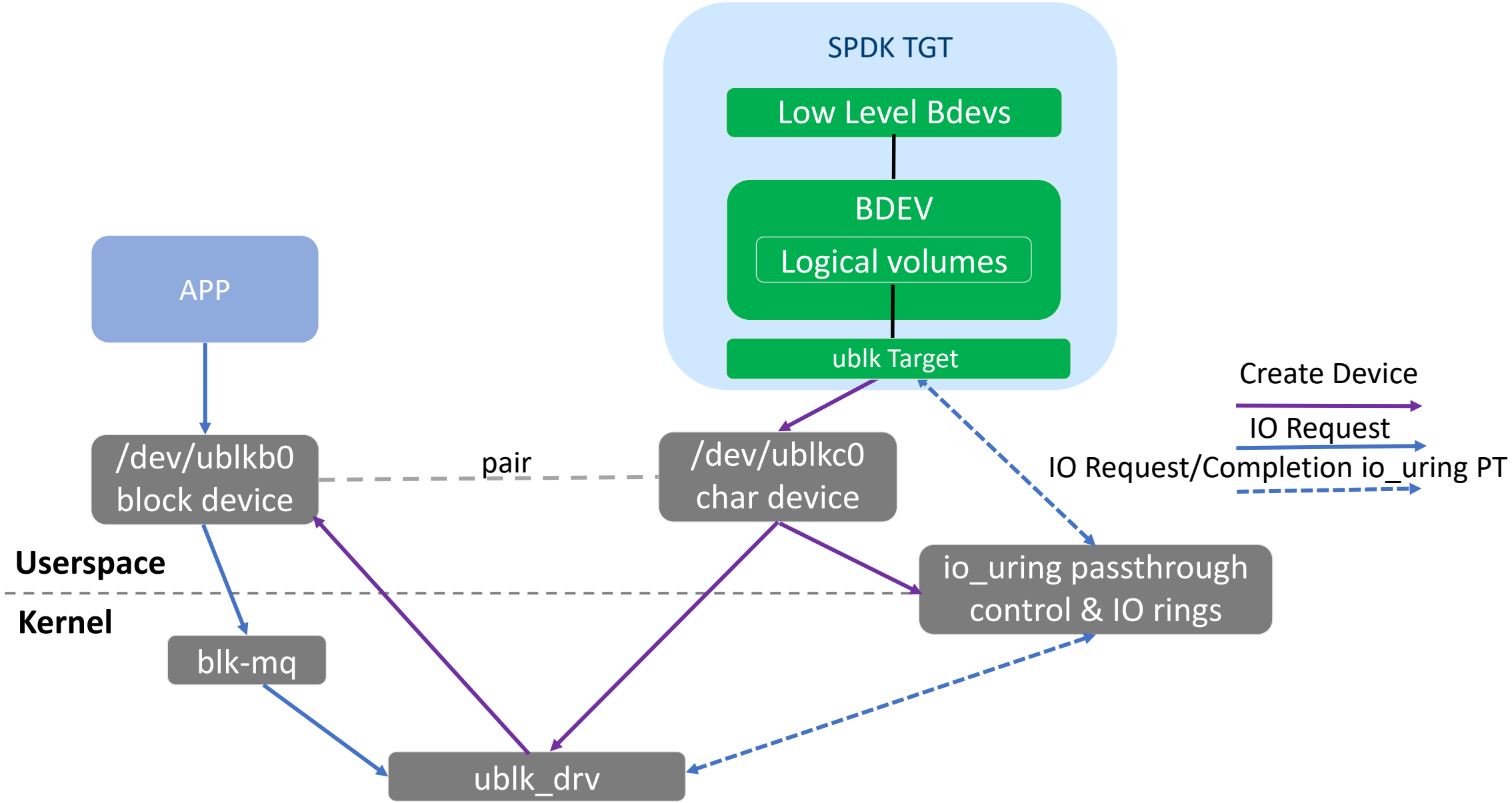
- Use io_uring to initialize, query and update device configuration
- Use io_uring to **exchange IO requests and completion**
- IO data is **copied once from page-cache into prefilled buffers.** Userspace backend prefills available buffers into kernel via io_uring.

- UBLK framework
  - Kernel module -- ublk_drv
  - Original userspace accessories:
    - libublksrv
    - ublk server – ublksrv



- SPDK can act as a ublk server – SPDK ublk target

UBLK


SPDK TGT
Low Level Bdevs
BDEV
Logical volumes
ublk Target
APP
/dev/ublkb0
block device
pair
/dev/ublkc0
char device
Create Device
IO Request
IO Request/Completion io_uring PT
Userspace
Kernel
io_uring passthrough
control & IO rings
blk-mq
ublk_drv

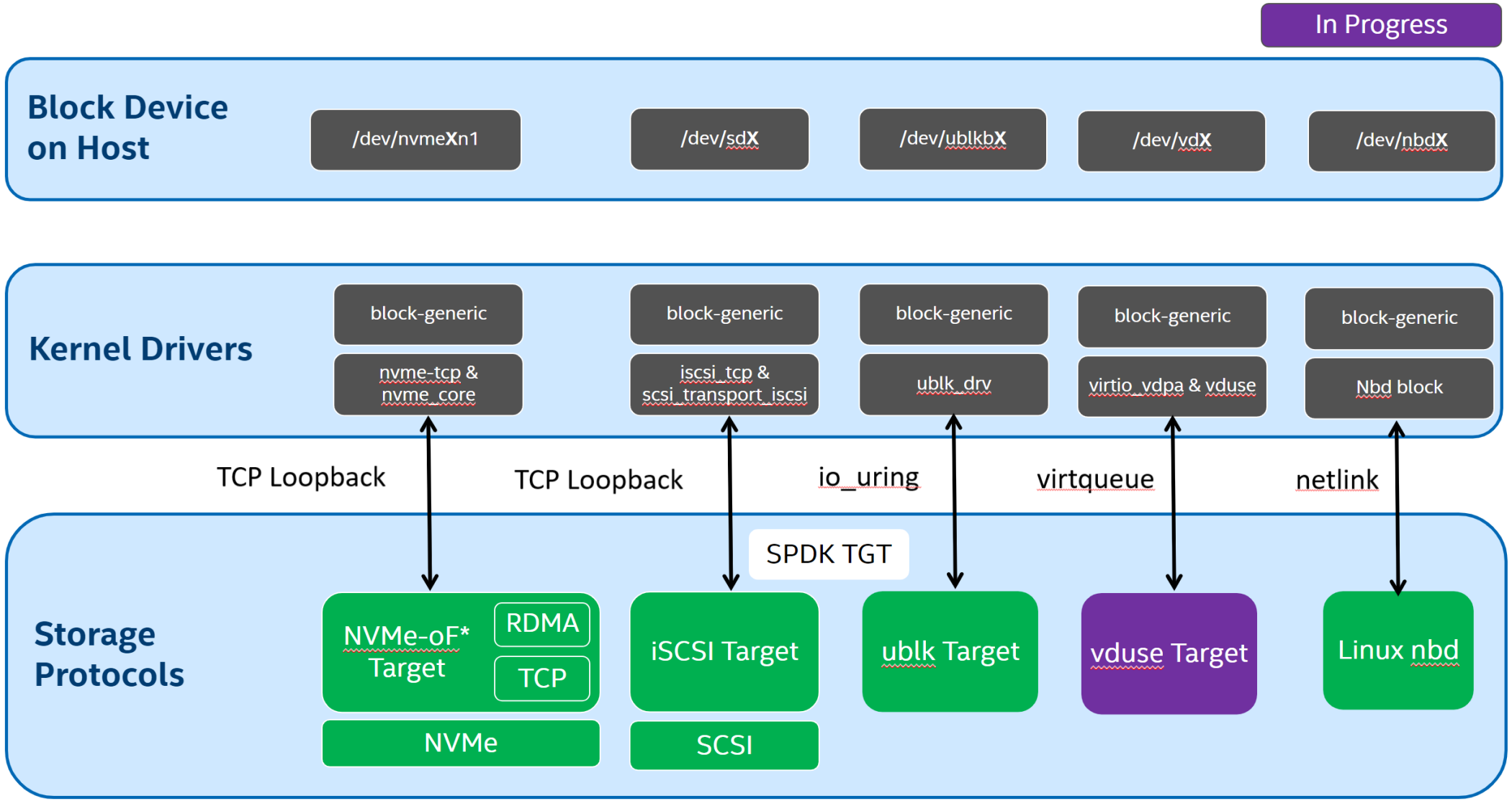
SPDK Progress

- SPDK ublk target is implemented as a high performance ublk server.
- SPDK ublk target is one of SPDK applications since v23.01.
  - No dependency with other modules in SPDK
- SPDK ublk target keeps the steps of `ublk_drv` kernel driver improvement.
  - USER COPY and GET FEATURES are already supported
  - Recovery feature is under development
- Visit https://spdk.io/doc/ublk.html for more information.

SPDK Progress

- VDUSE was evaluated earlier with good result.
  - Visit SPDK VDUSE concept evaluation: SPDK VDUSE backend
- Still not upstreamed into SPDK master branch
  - DPDK dependency
  - Some duplicated code with vhost library
  - Further abstraction together with vhost target
- Expect more progress to upstream
  - There has finally been progress for VDUSE support in this DPDK

SPDK Progress



SPDK Progress

- Low Level BDEVs
  - BDEV module creates abstraction layer that provides common API for all devices
  - Supports RAID/Linux AIO/Ceph RBD/iSCSI/NVMe etc.
  - Custom BDEVs can be added into SPDK BDEV framework easily
  - Provides also vbdev modules which creates block devices on existing bdev, Logical Volume is using this mechanism
- SPDK Logical Volume
  - Flexible storage space management library
  - Features like thin provisioning/snapshots/external snapshots/clone are supported

Now, with storage protocols such as NVMe-oF/iSCSI/Ublk and BDEV service, SPDK is great data path for cloud native block storage solution.

Comparison

- Software stack
  - VDUSE relies on a thick virtio/vdpa dependency
    - Virtio-blk
    - Virtio
    - Virtio-vdpa
    - Vdpa
    - VDUSE
  - ublk is thin dependency
    - ublk_drv -- Self-contained. Only depends on io_uring

Comparison

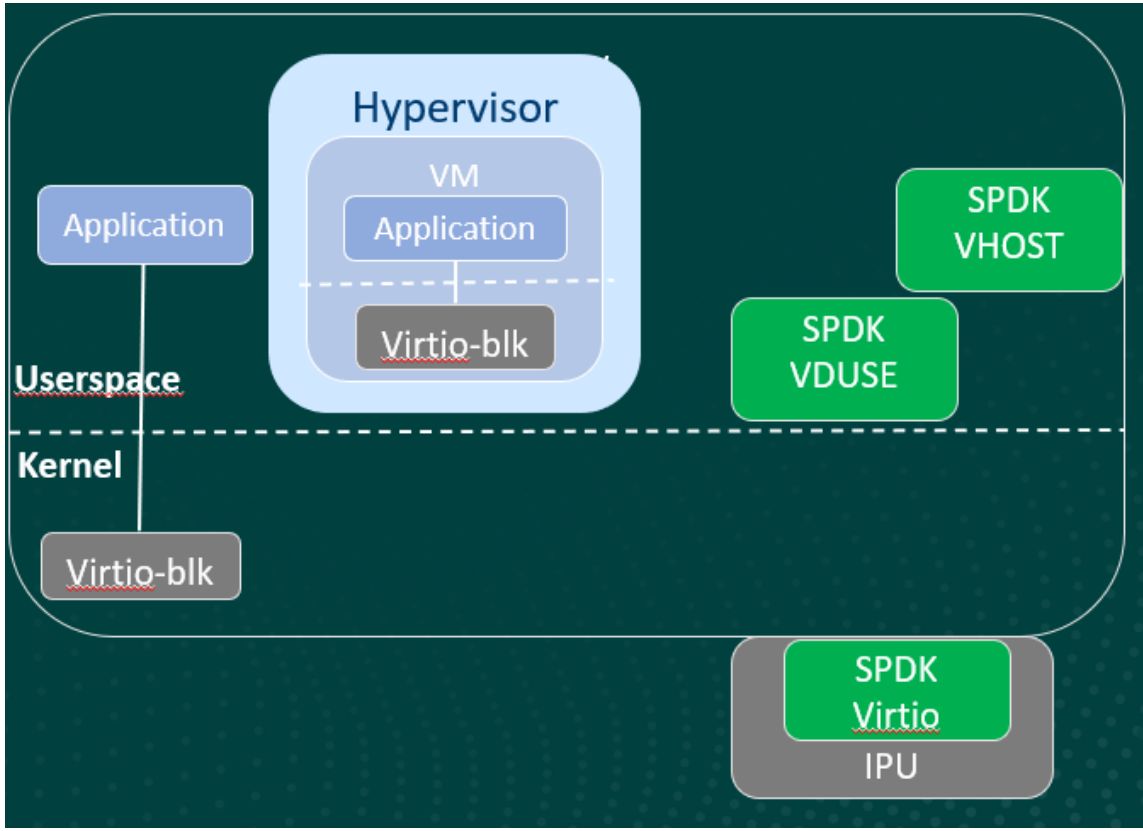
- Maturity
  - VDUSE
    - High code quality with strict and conscientious review/revise
    - Proven in ByteDance production environment
    - Raised in 2020 and merge started in 2021
  - ublk
    - Raised and merged in 2022
    - Require more workloads' verification
    - Turn quickly and get improved continuously

Comparison

- Ecosystem
  - VDUSE – virtio ecosystem
    - Reuse the whole facilities built for virtio
    - Easy to be extended for other virtio semantics, like filesystem semantic – virtio-fs
  - ublk
    - Compatible to Linux block and io_uring settings.
    - Specific to block service
    - Flexible to adapt new io_uring feature or add new ublk command definitions.

Comparison

- Compatibility
  - VDUSE – expose virtio-blk interface
    - Virtio-blk is implementable for both software and hardware
    - Virtio-blk is adopted in both virtualization and bare metal
    - VDUSE gives users a consistent frontend interface among VM/Container/IPU
  - ublk
    - Expose high performance generic block device for localhost



Comparison

Efficiency and Performance

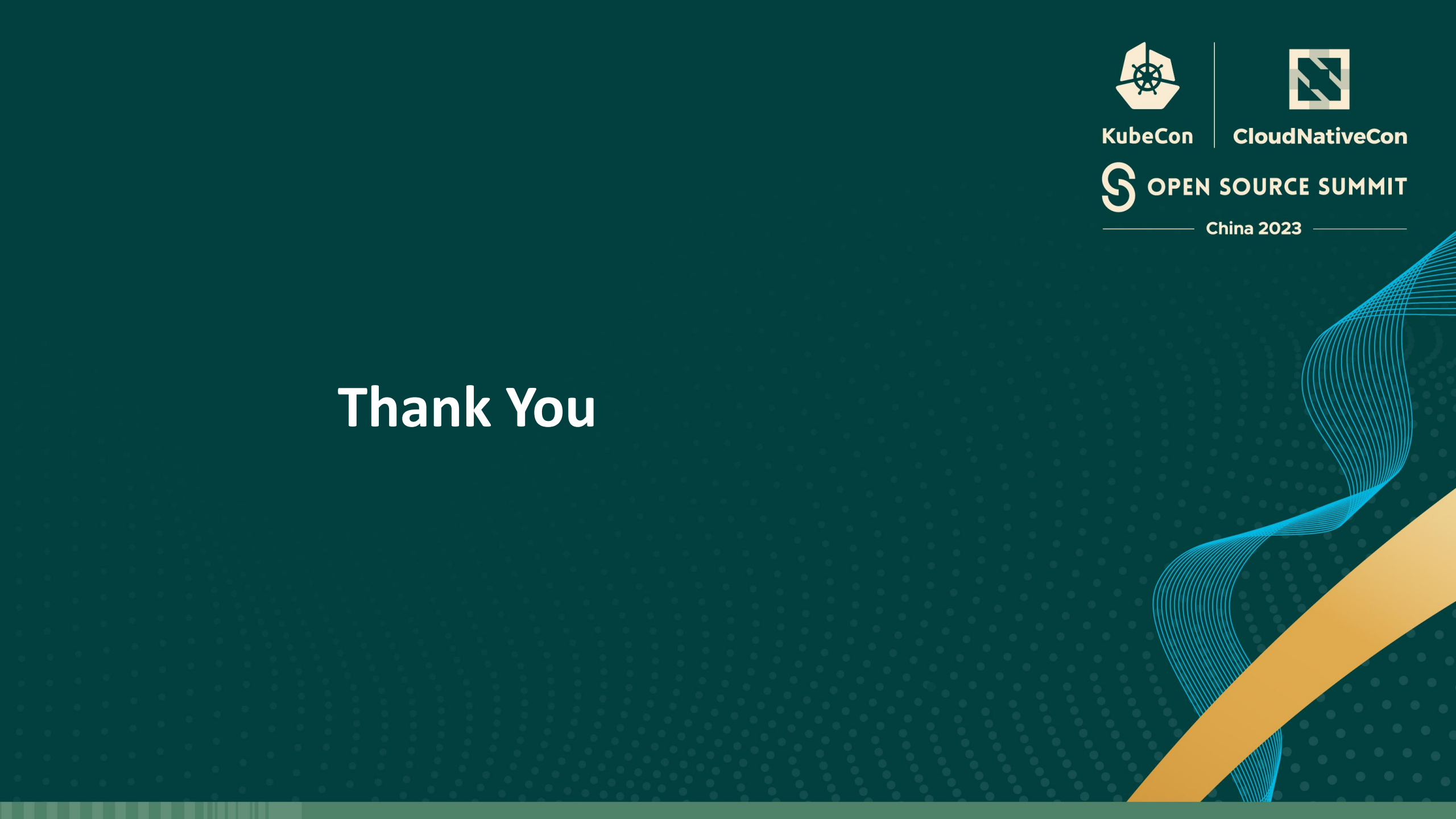
- Both VDUSE and ublk have a thin and efficient data path
- VDUSE
  - Virtqueue
  - Shared memory
- ublk
  - io_uring
  - Buffer prefill or buffer allocation via another round io_uring cmd
- Performance observed: ublk is better when counting overall cpu usage
  - But gap is not so large as described in kernel maillist
  - VDUSE continues improvement on IRQ Callback Affinity

Comparison

- Cooperability with SPDK
  - VDUSE -- Great single reactor performance
    - Thread modeling same with vhost-blk
    - Data copy is conducted by kernel kworker threads out of SPDK context
  - ublk – Better overall performance, but
    - ublk limits pthread context switch which blocks spdk_thread scheduling
    - Data copy is conducted within SPDK context – poor single reactor performance

Conclusion

- ublk and VDUSE are emerging modern solutions for userspace block services, especially for SPDK
- The SPDK implementations of ublk and VDUSE both show great performance improvement over iSCSI and NVMe-oF
- ublk and VDUSE show specific advantages separately


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Thank You