Introduce a SPDK vhost FUSE target to accelerate File Access in VMs and containers

Changpeng Liu, Xiaodong Liu

Cloud Storage Software Engineer
Intel Data Center Group

Notices & Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

No computer system can be absolutely secure.

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 http://www.intel.com/benchmarks .

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 factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks .

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.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Storage Performance Development Kit

Available via spdk.io

@SPDKProject

Scalable and Efficient Software Ingredients

- User space, lockless, polled-mode
- Up to millions of IOPS per core
- Minimize average and tail latencies
- Designed for non-volatile media

Storage Reference Software

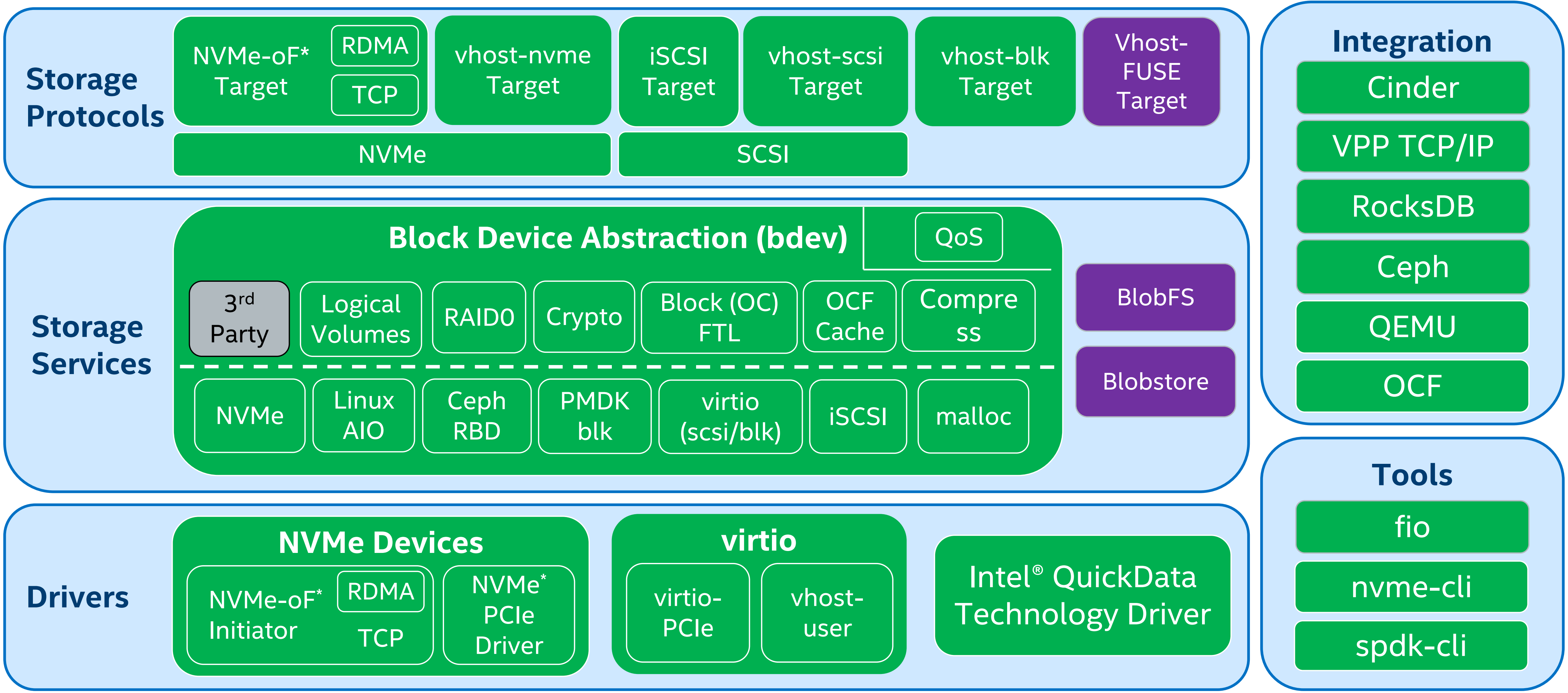
- Optimized for latest generation CPUs and SSDs
- Provides Future Proofing
- Extends to Storage Virtualization and Networking

Open Source community

- Open source building blocks (BSD licensed)
- Faster TTM, fewer resources required

Vhost-FUSE

SPDK ARCHITECTURE



Storage Protocols

- NVMe-oF* Target (RDMA, TCP)
- vhost-nvme Target
- iSCSI Target
- vhost-scsi Target
- vhost-blk Target
- Vhost-FUSE Target
- NVMe
- SCSI

Storage Services

Block Device Abstraction (bdev)

QoS

- 3rd Party
- Logical Volumes
- RAID0
- Crypto
- Block (OC) FTL
- OCF Cache
- Compress
- NVMe
- Linux AIO
- Ceph RBD
- PMDK blk
- virtio (scsi/blk)
- iSCSI
- malloc

- BlobFS
- Blobstore

Drivers

NVMe Devices

- NVMe-oF* Initiator (RDMA, TCP)
- NVMe* PCIe Driver

virtio

- virtio-PCIe
- vhost-user

Intel® QuickData Technology Driver

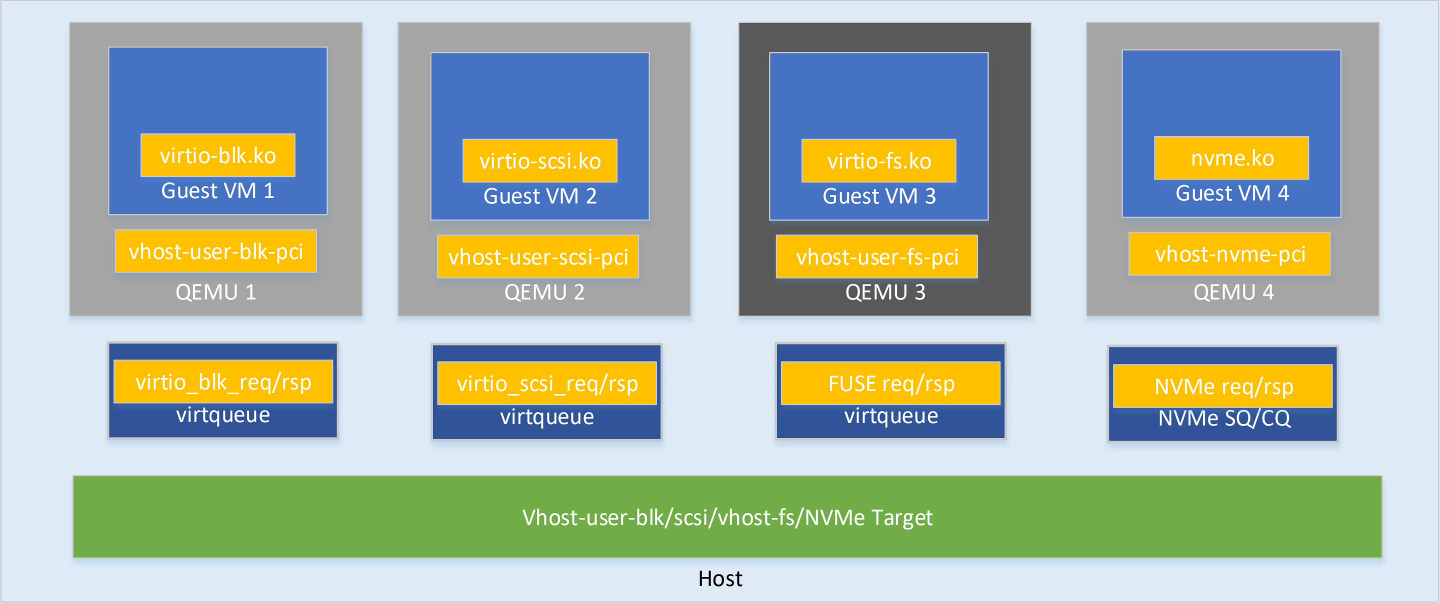
Integration

- Cinder
- VPP TCP/IP
- RocksDB
- Ceph
- QEMU
- OCF

Tools

- fio
- nvme-cli
- spdk-cli

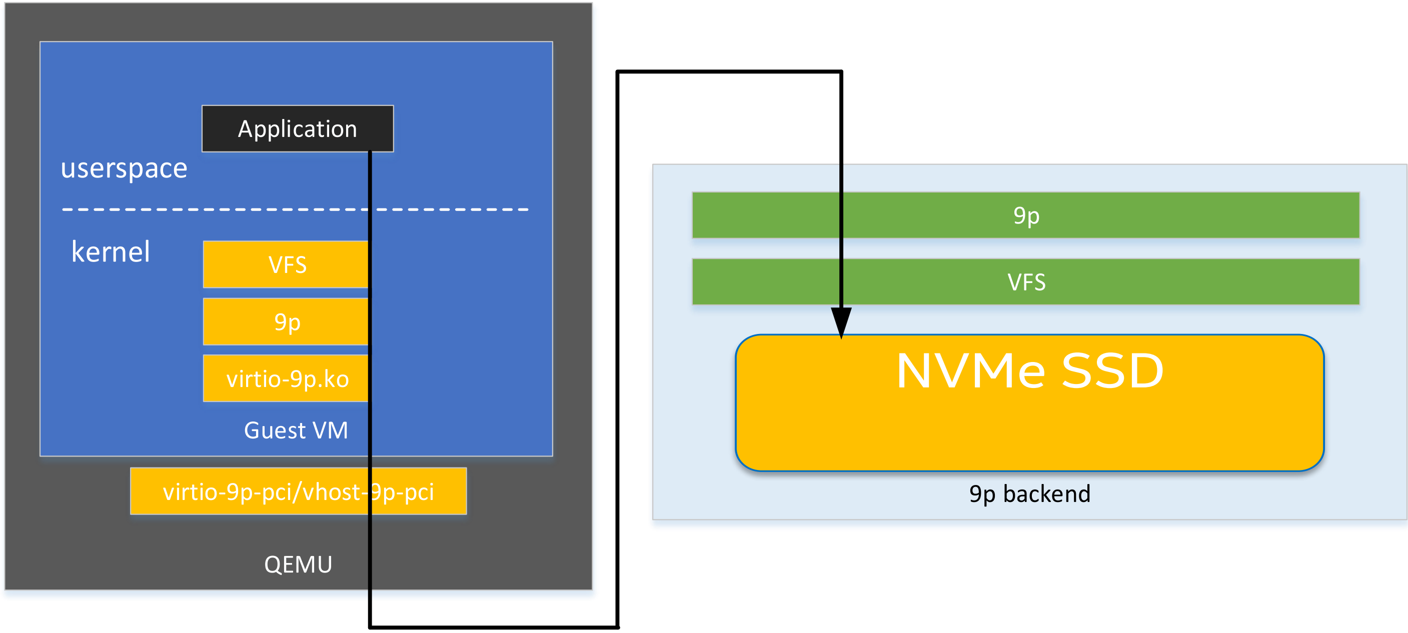
Unified SPDK Vhost user target



- SPDK vhost target can provide virtio-blk/virtio-scsi/virtio-fs/NVMe in Guest.
- Build on common vhost-user library to parse vring and NVMe queues.
- Eliminate MMIO access for submission with polling mechanism.

Sharing Host Files with the Guest

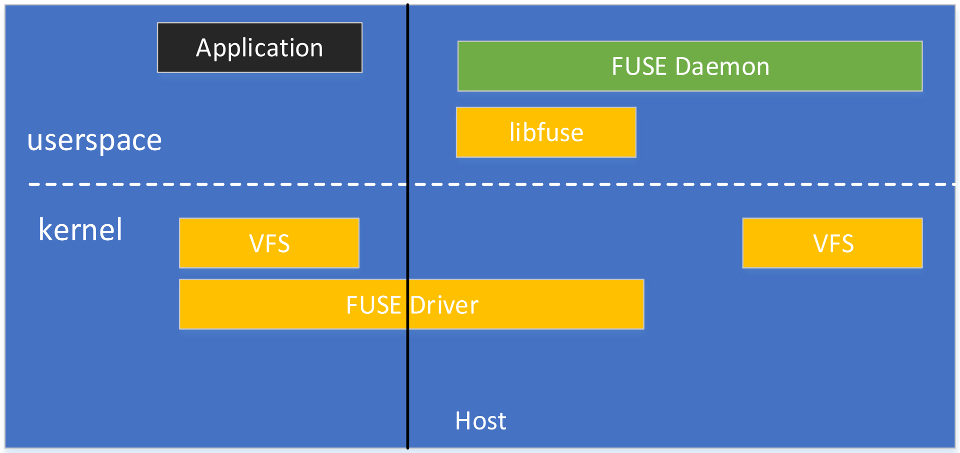
- Plan 9 folder sharing over Virtio - I/O virtualization
- QEMU can process the 9p request and send to backend via local existing POSIX APIs.
- Another optimization for Clear Containers which moves the IO process into the Host kernel space, eliminate syscall from usersapce to kernel space compared with previous solution.



Container deployments utilize explicit or implicit file sharing between host filesystem and containers.

What’s FUSE

- FUSE (Filesystem in Userspace) is an interface for userspace programs to export a filesystem to the Linux kernel. The FUSE project consists of two components: the fuse kernel module and the libfuse userspace library. libfuse provides the reference implementation for communicating with the FUSE kernel module.
- How can FUSE be used in virtualization environment and accelerate shared file access between different VMs?



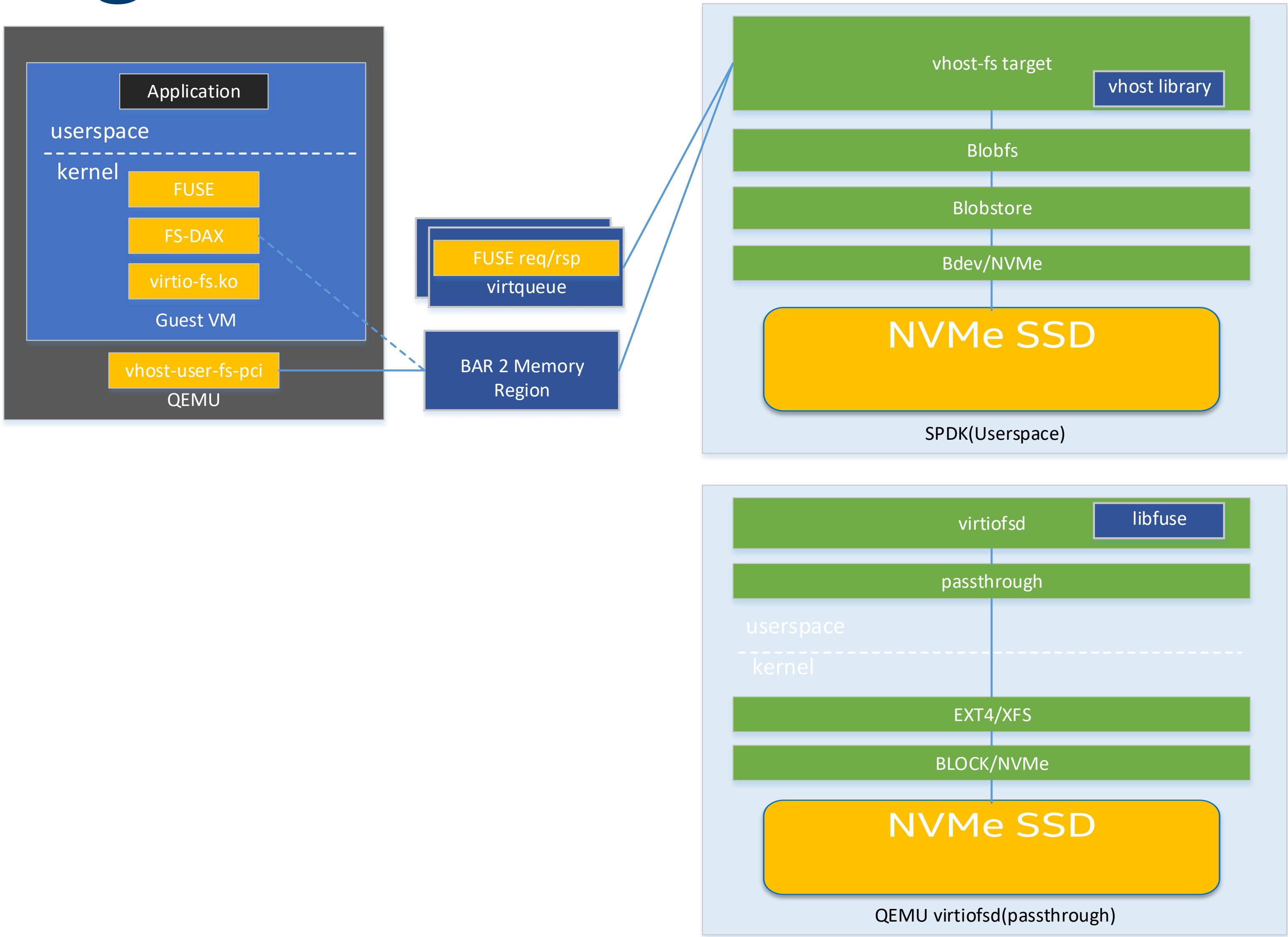
Example usage of FUSE

Introduction to Virtio-fs

- Virtio-fs is a shared file system that lets virtual machines access a directory tree on the host. Unlike existing approaches, it is designed to offer local file system semantics and performance. This is especially useful for lightweight VMs and container workloads, where shared volumes are a requirement.
- Virtio-fs was started at Red Hat and is being developed in the Linux, QEMU, FUSE, and Kata Containers communities that are affected by code changes.
- Virtio-fs uses FUSE as the foundation. A VIRTIO device carries FUSE messages and provides extensions for advanced features not available in traditional FUSE.
- DAX support via virtio-pci BAR from host huge memory.

SPDK Vhost-fs Target vs. QEMU Virtiofsd

- Eliminate userspace/kernel space context switch by providing a user space file system.
- Zero copied for both READ and WRITE.
- Hugetlbfs is required.
- DAX(Direct Access for files)

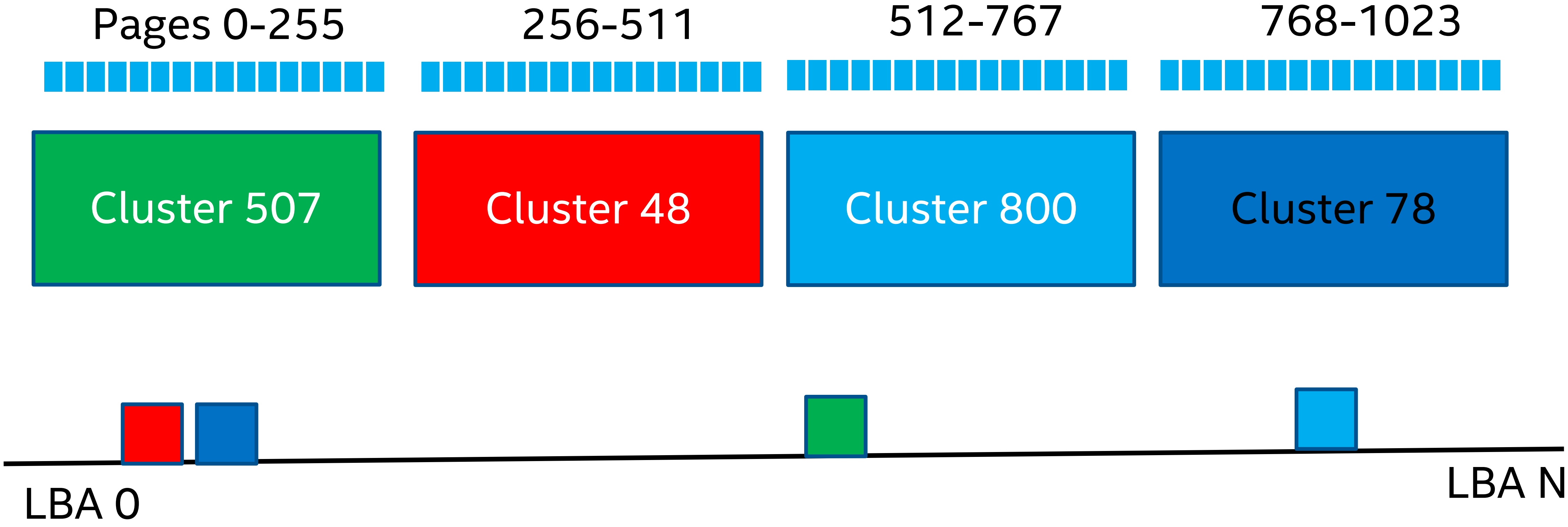


Introduction to SPDK Blobfs/Blobstore

- Application interacts with chunks of data called blobs:
- Designed for application that don’t consume block, such as Rocksdb.
- Designed for fast media, such as NVMe SSDs.
- Mutable array of pages of data, accessible via ID.
- Asynchronous, no blocking, queuing or waiting.
- Fully parallel, no locks in IO path.
- Data is direct, Metadata is cached, minimal support for xattrs.
- Logical volumes with snapshot and thin provisioning.

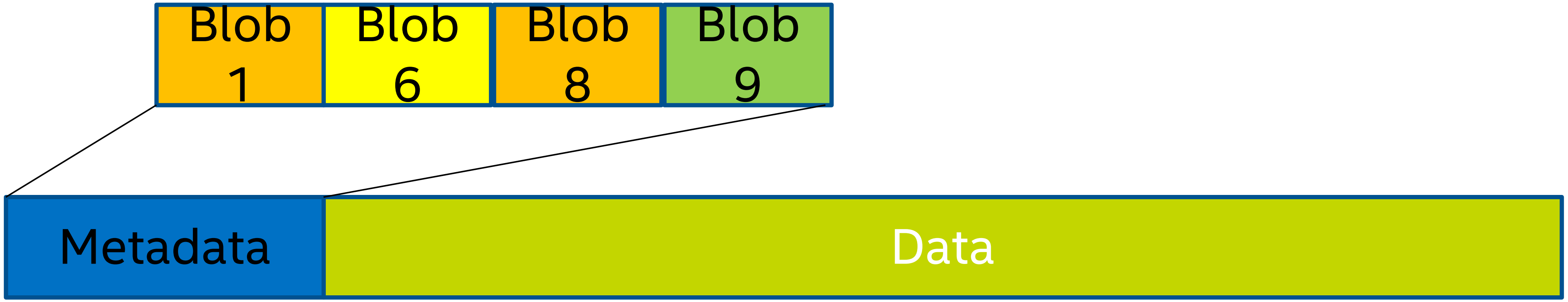
Blobstore Design - Layout

- A blob is an array of pages organized as an ordered list of clusters



Blobstore Design - Metadata

- Stored in pages in reserved region of disk
- Not shared between blobs
- One blob may have multiple metadata pages



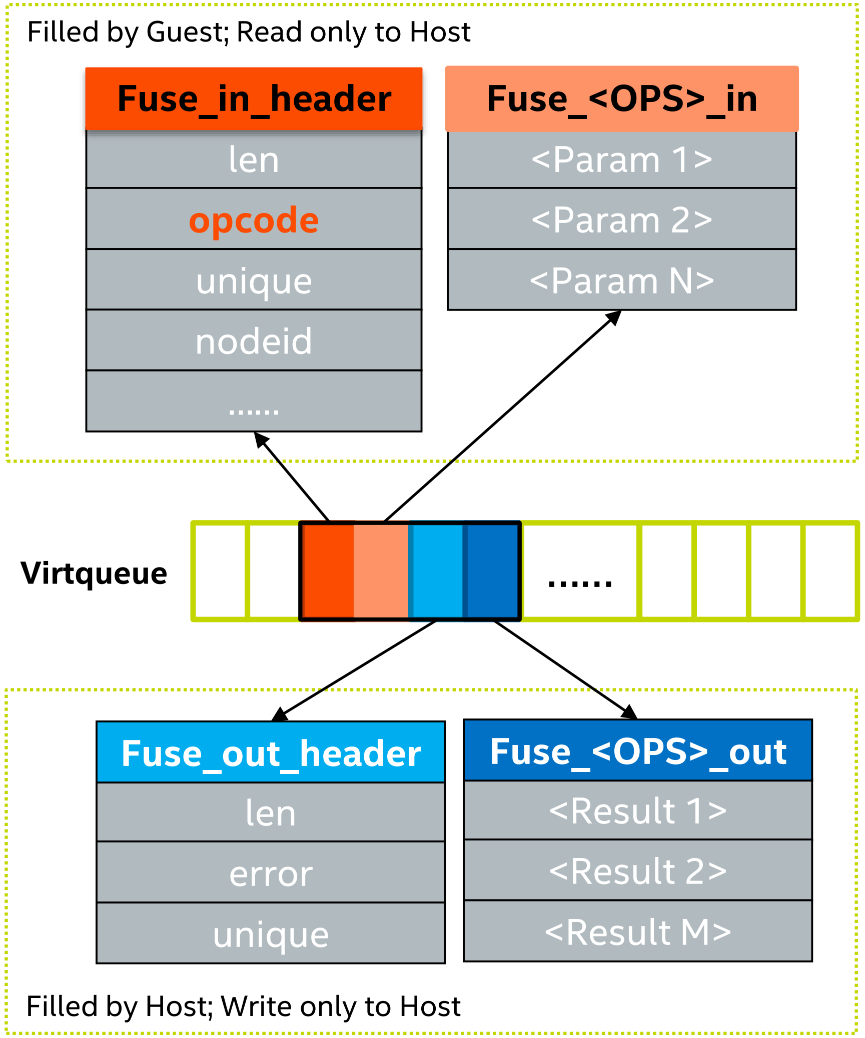
SPDK Blobfs API vs. FUSE

- Open, read, write, close, delete, rename, sync interface to provide POSIX similar APIs.
- Asynchronous APIs are also provided.

FUSE Command	Blobfs API
Lookup	spdk_fs_iter_first, spdk_fs_iter_next
Getattr	spdk_fs_file_stat_async
Open	spdk_fs_open_file_async
Release	spdk_file_close_async
Create	spdk_fs_create_file_async
Delete	spdk_fs_delete_file_async
Read	spdk_file_readv_async
Write	spdk_file_writev_async
Rename	spdk_fs_rename_file_async
Flush	spdk_file_sync_async

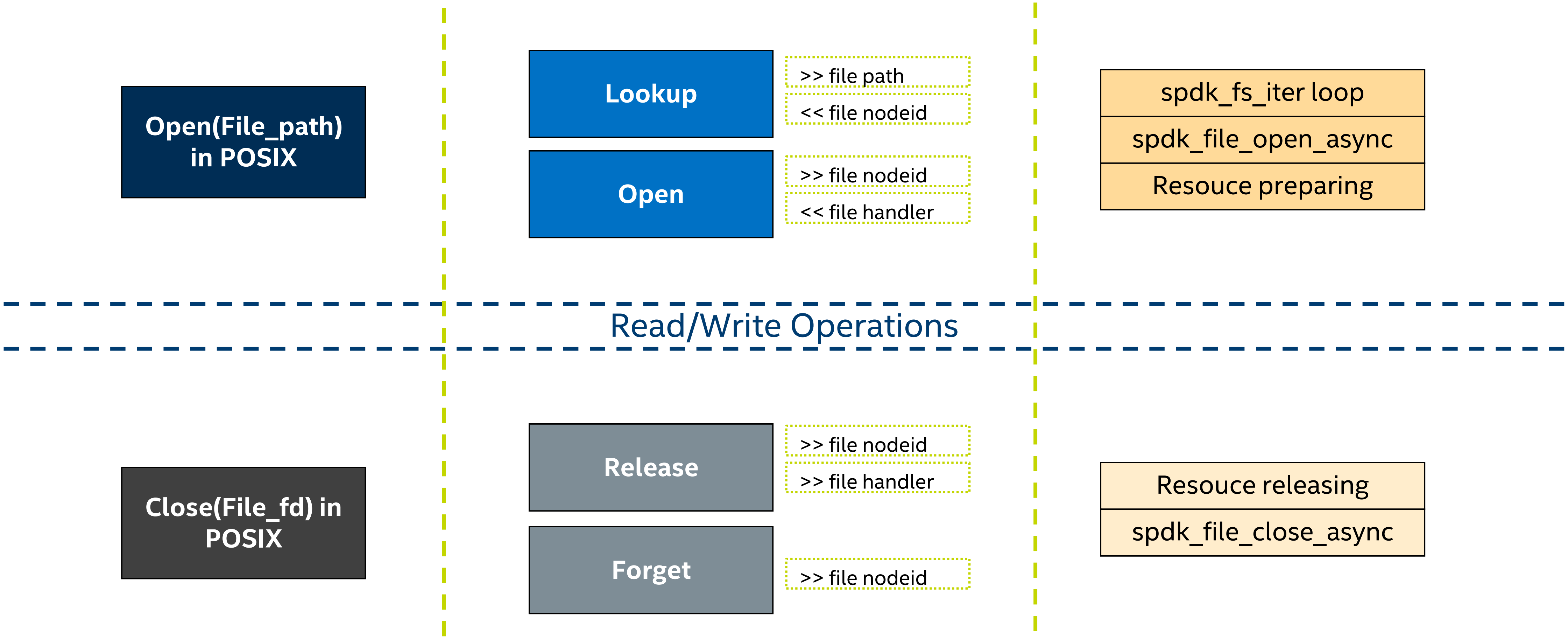
Operation Mapping of Fuse in Virtqueue

- General FUSE command has 2 parts: request and response.
- General FUSE request is consisted with IN header and operation specific IN parameters.
- General FUSE response is consisted with OUT header and operation specific OUT results.


Filled by Guest; Read only to Host

Filled by Host; Write only to Host

Open and Close Operations in Fuse and SPDK



Open(File_path) in POSIX

Lookup

>> file path

<< file nodeid

Open

>> file nodeid

<< file handler

spdk_fs_iter loop

spdk_file_open_async

Resouce preparing

Read/Write Operations

Close(File_fd) in POSIX

Release

>> file nodeid

>> file handler

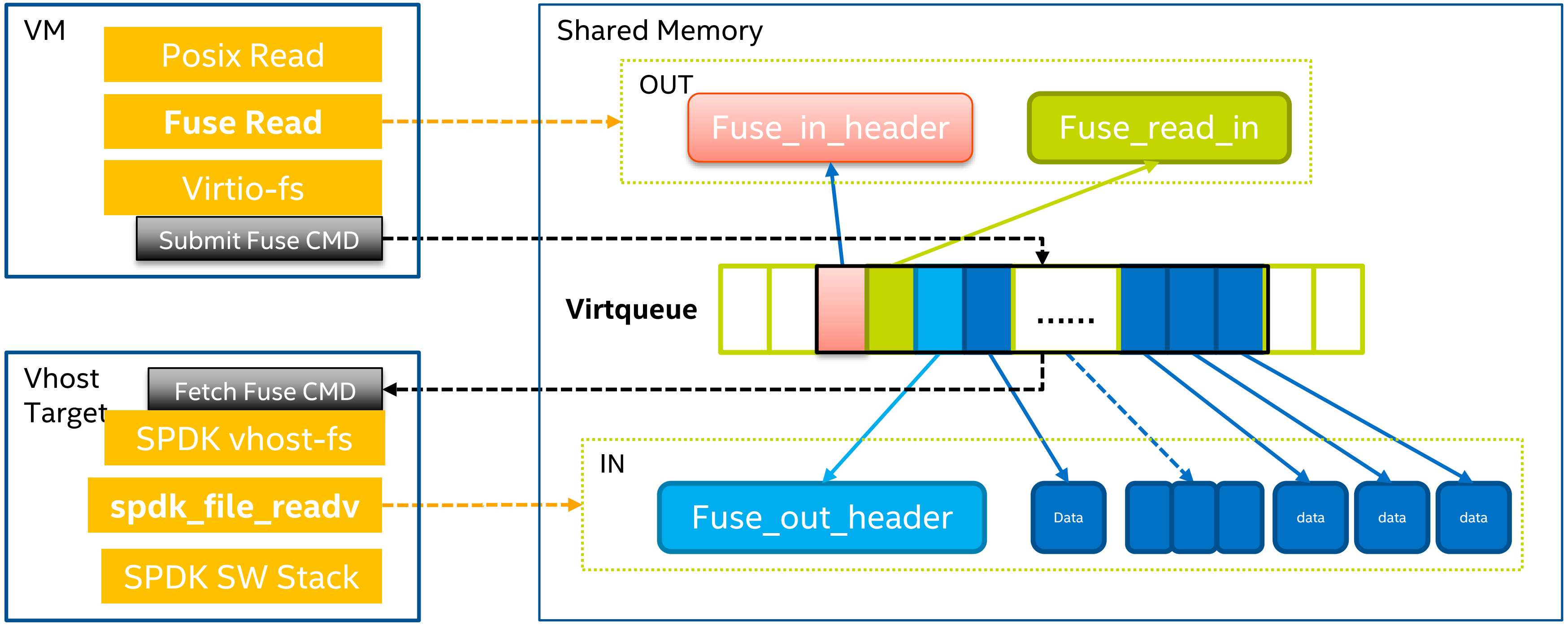
Forget

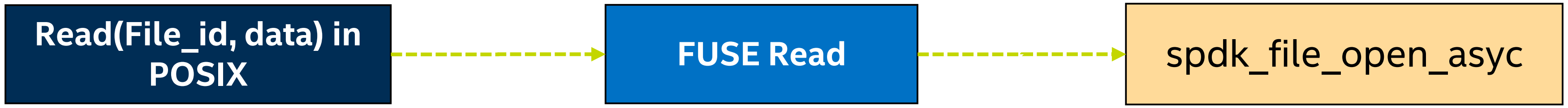
>> file nodeid

Resouce releasing

spdk_file_close_async

Implementation Details with Read/Write





Summary & Future plan

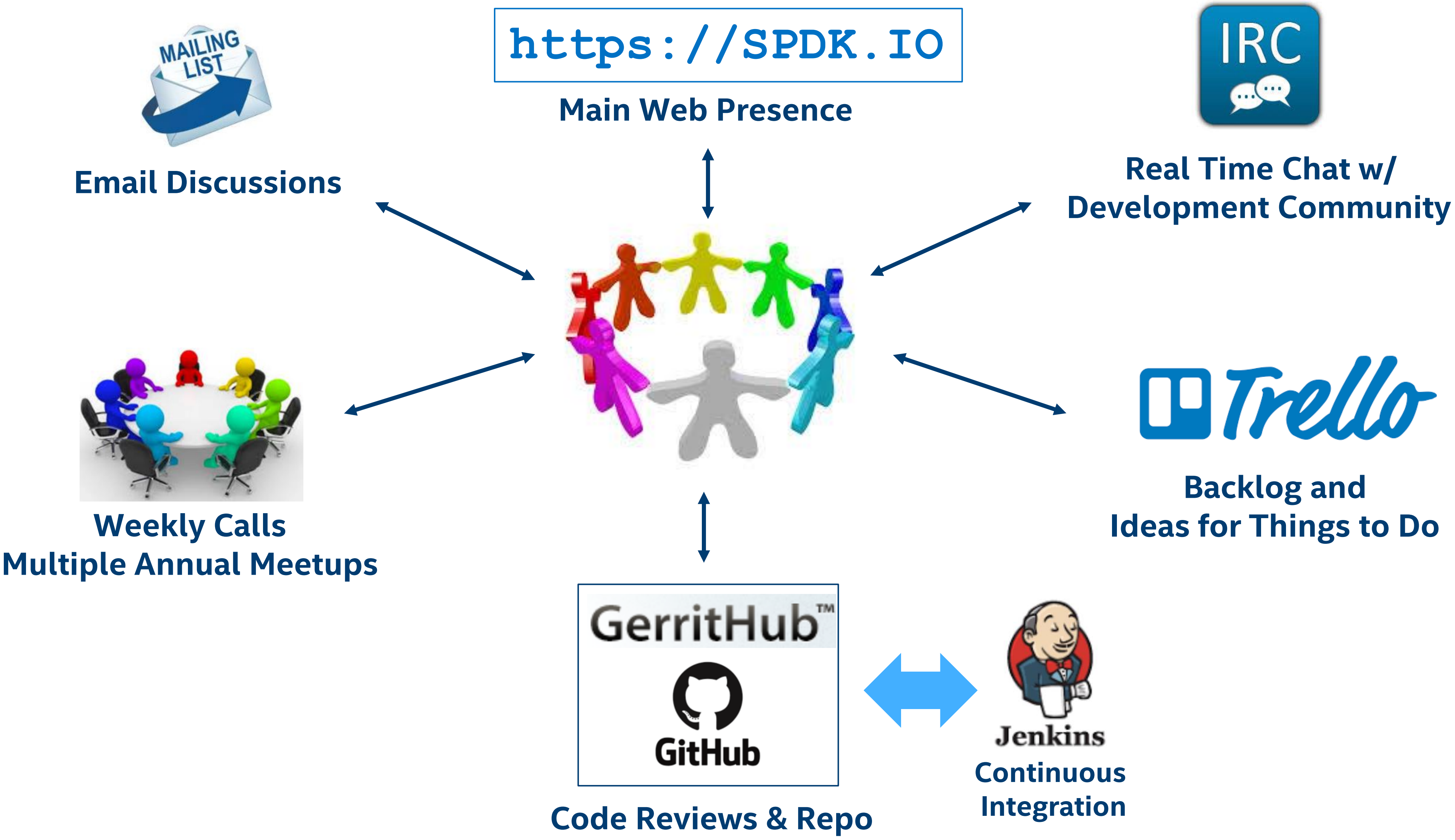
- **Summary**
- SPDK blobfs can support limited file APIs, and only append write is supported for now.
- Friendly for WRITE workload and simple READ cache feature is enabled.
- Optimized for big files, and not friendly for small files.

- **Future plan**
- Continue to improve existing SPDK blobfs, include the thread model as well as asynchronous APIs.
- Benchmarks.

Useful links:

1. https://review.gerrithub.io/#/c/spdk/spdk/+/449162/
2. https://review.gerrithub.io/#/c/spdk/spdk/+/449163/
3. https://virtio-fs.gitlab.io/

SPDK Community



https://SPDK.IO

Main Web Presence

Email Discussions

Real Time Chat w/ Development Community

Weekly Calls
Multiple Annual Meetups

Backlog and
Ideas for Things to Do

Code Reviews & Repo

Continuous Integration

intel®

experience
what's inside™