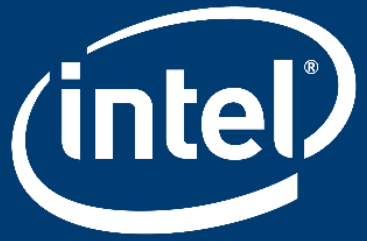


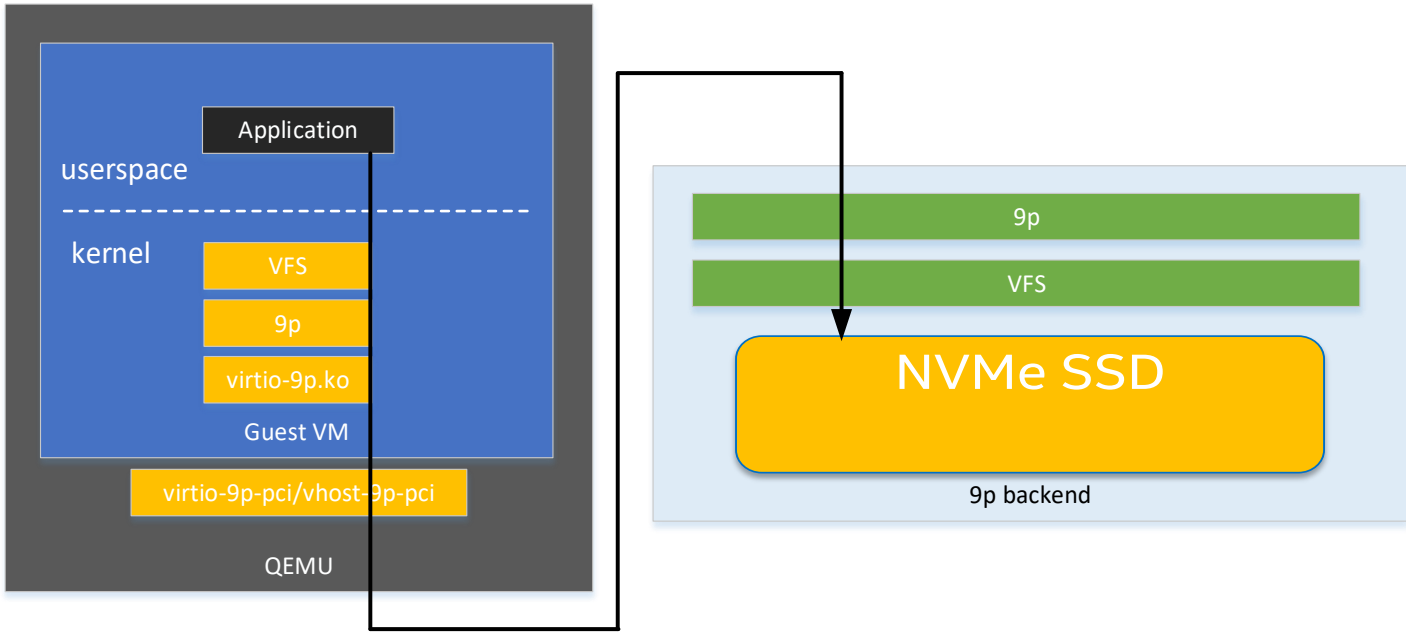
Introduce a SPDK vhost FUSE target to accelerate File Access in VMs and containers

Changpeng Liu, Xiaodong Liu

Cloud Storage Software Engineer
Intel Data Center Group

Sharing host files with the Guest

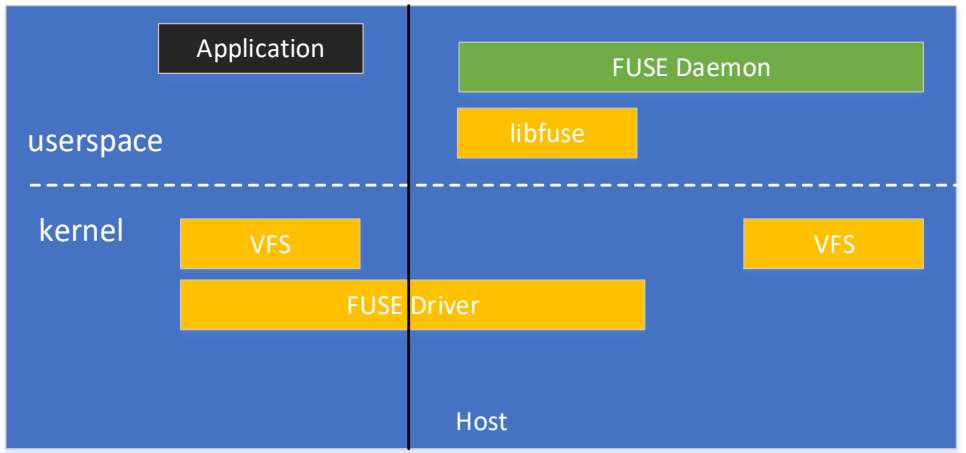
- Plan 9 folder sharing over Virtio - I/O virtualization
- QEMU can process the 9p request and send to backend via local existing POSIX APIs.
- Another optimization for clearcontainers which moves the IO process into the Host kernel space, eliminate syscall from usersapce to kernel space compared with 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 FUSE can it be used in virtualization environment and accelerate shared file access between different VMs?



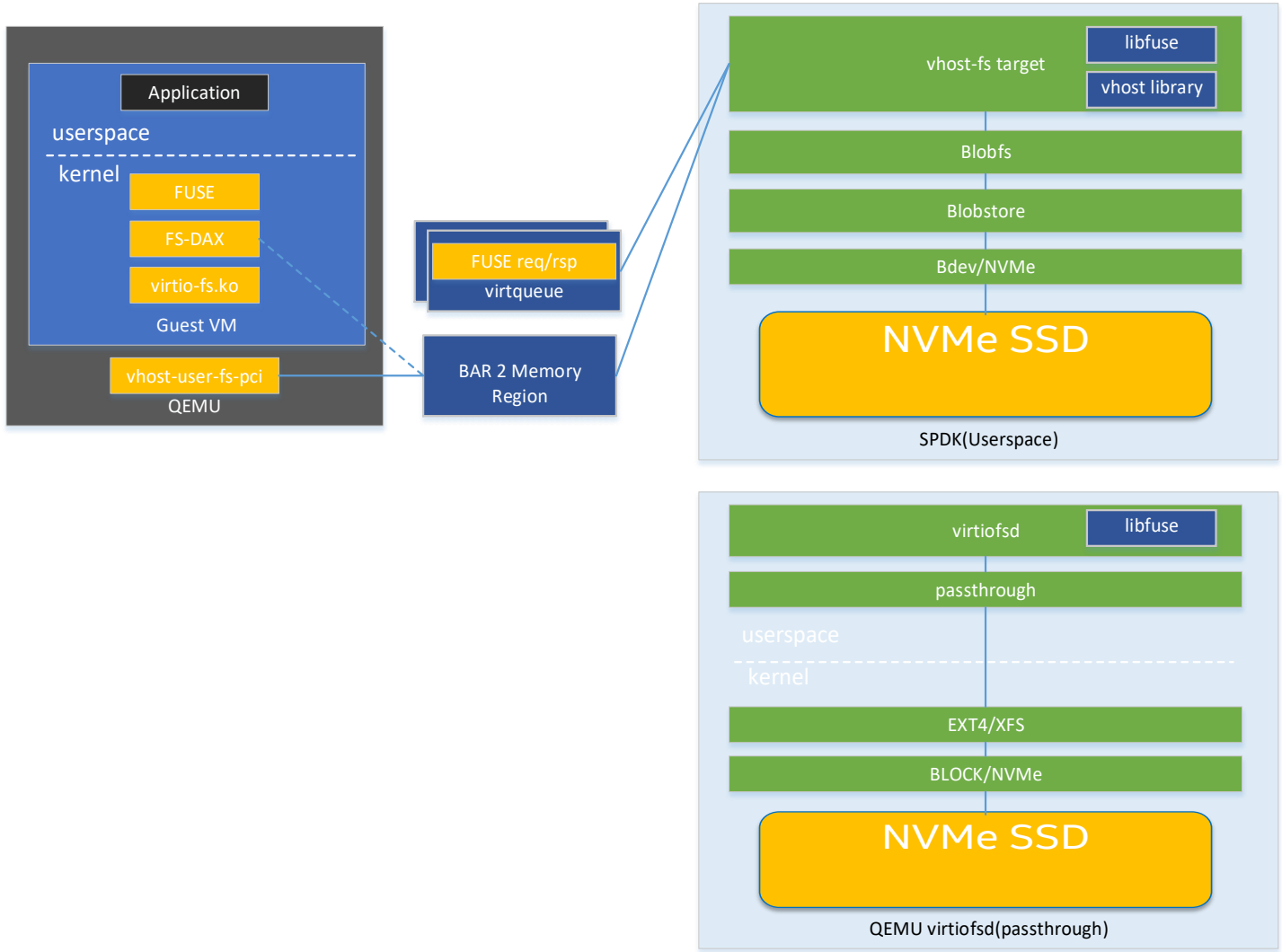
Example usage of FUSE

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 viriofsd

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

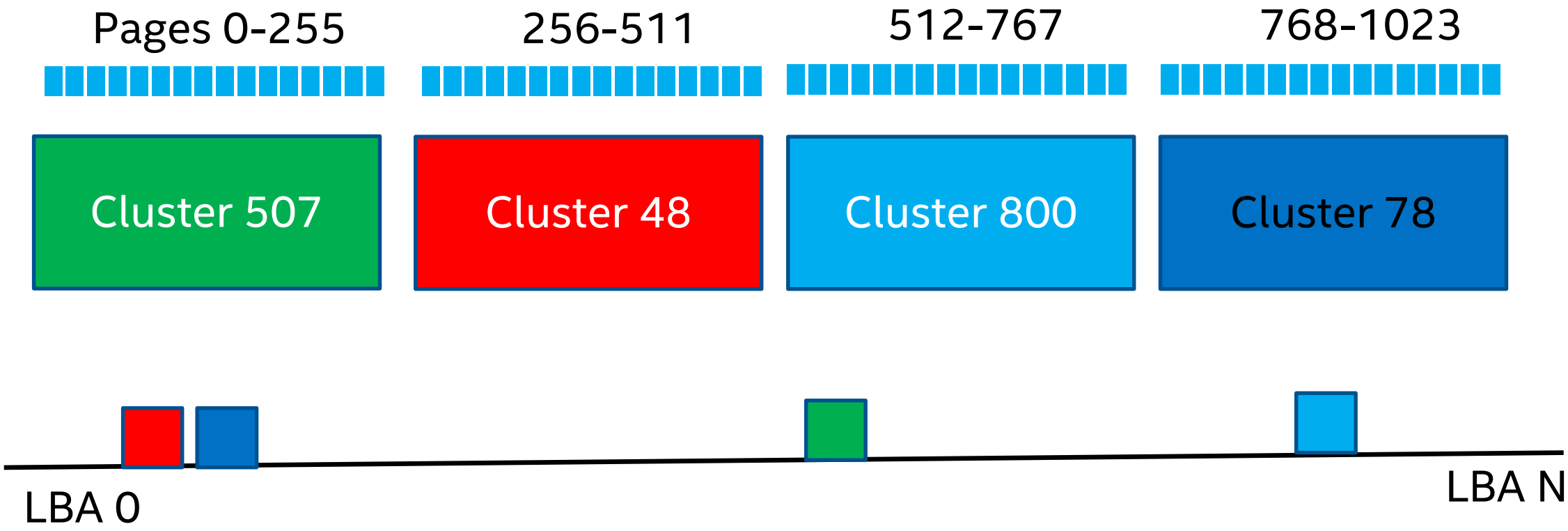


Introduction to SPDK Blobfs/Blobstore

- 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.

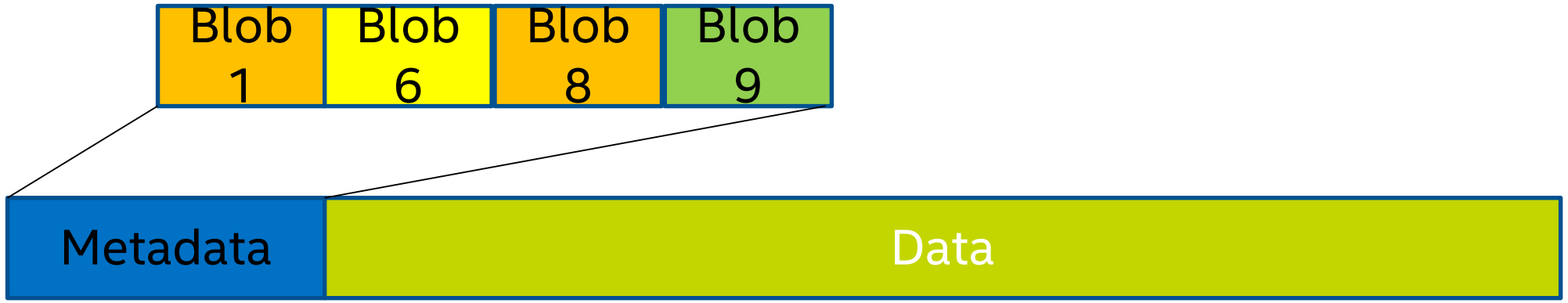
Blobstore Design - Layout

- 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

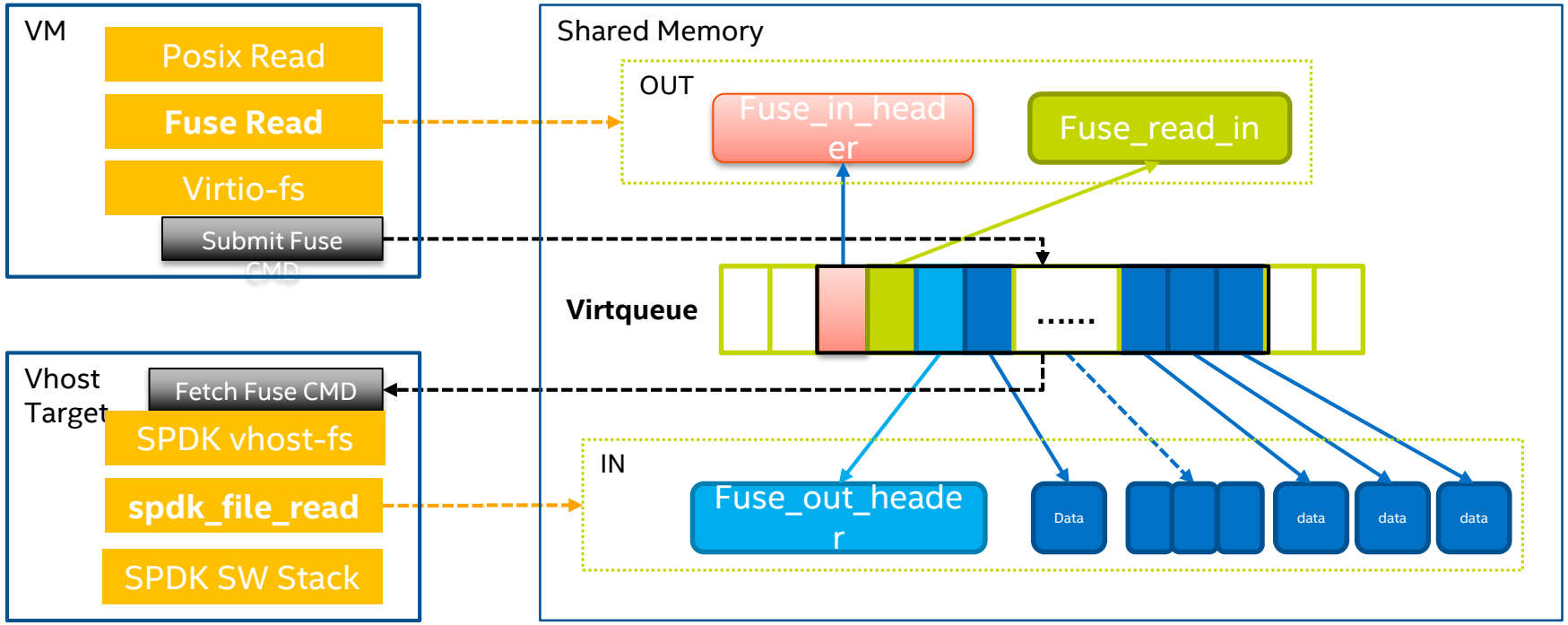


SPDK Blobfs API vs. FUSE

- 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
Open	spdk_fs_open_file
Release	spdk_file_close
Create	spdk_fs_create_file
Delete	spdk_fs_delete_file
Read	spdk_file_readv
Write	spdk_file_writev
Rename	spdk_fs_rename_file
Flush	spdk_file_sync

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/


intel
experience
what's inside™