virtio-fs在dpu场景中的机会与挑战

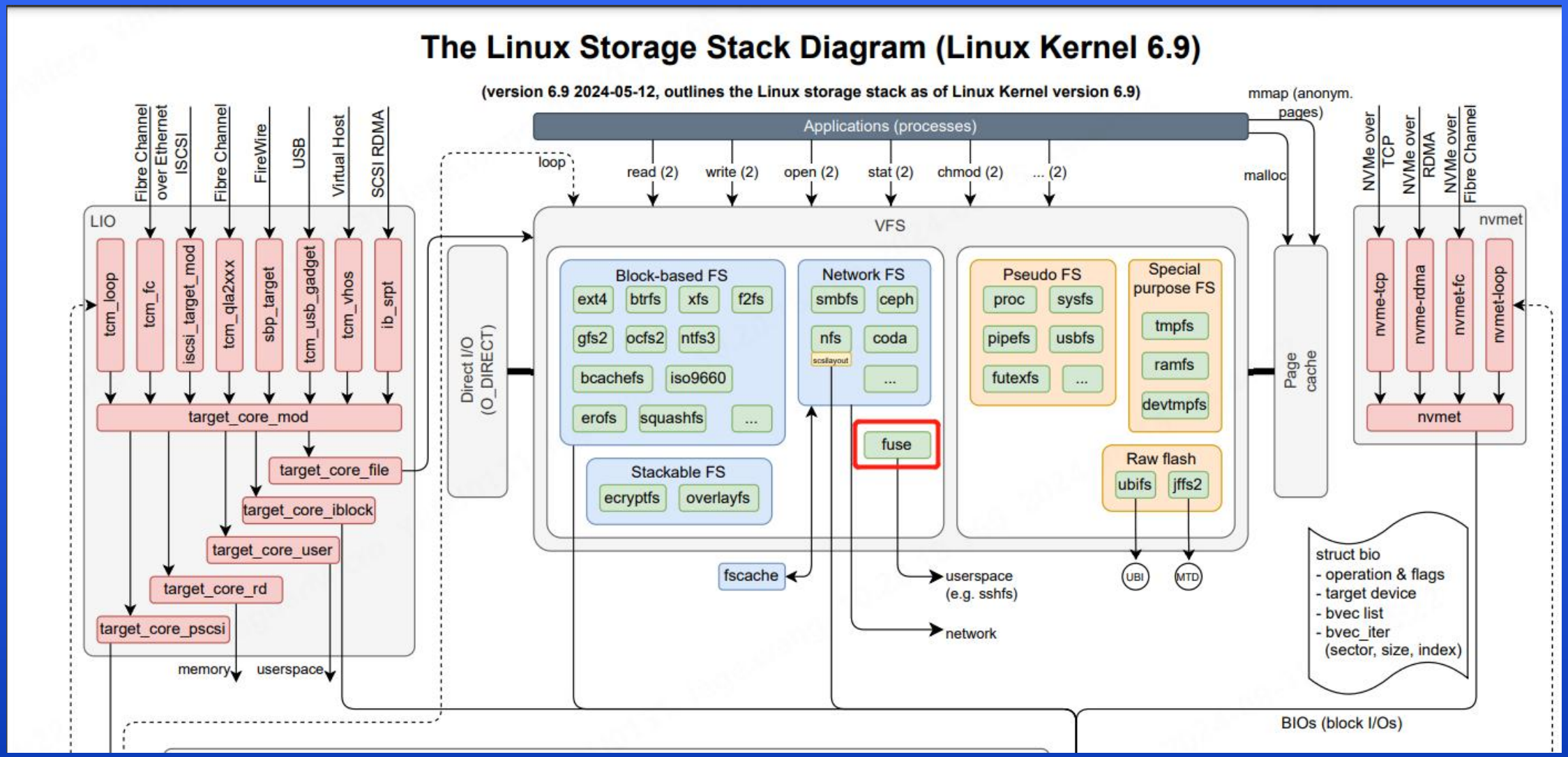
陈安庆 云豹智能软件系统首席架构师

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler

目录

- fuse & virtio-fs introduction
- virtio-blk vs virtio-fs
- corsica virtio-fs卸载方案
- 性能&收益
- Todo lists

fuse & virtio-fs introduction – 内核存储栈fuse层次



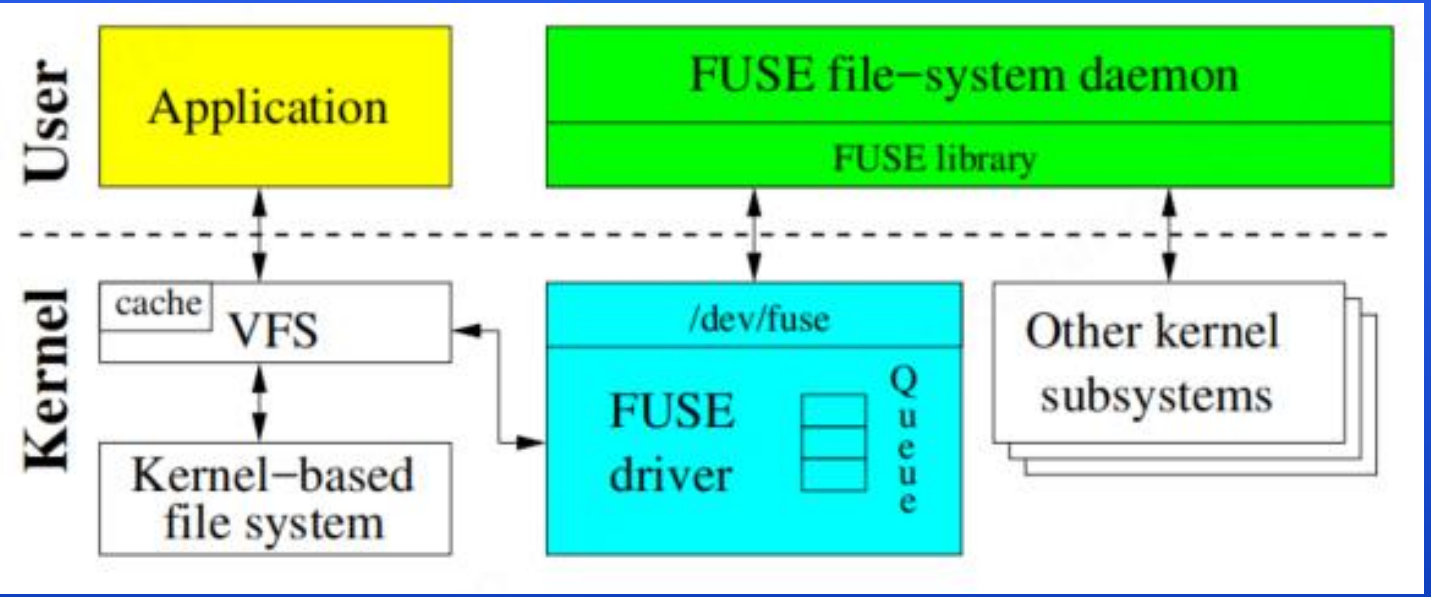
https://www.thomas-krenn.com/de/wikiDE/images/a/ad/Linux-storage-stack-diagram_v6.9.pdf

fuse & virtio-fs introduction – fuse架构

FUSE: Filesystem In User space：一种用于实现用户态文件系统的框架，包含两个核心组件：

- fuse内核模块，位于Linux内核的fs/fuse目录
- 用户态文件系统daemon，实现文件系统的业务逻辑，通常基于libfuse库

fuse内核模块与用户态文件系统服务之间依靠/dev/fuse字符设备进行FUSE请求的处理及响应。



图片摘自论文： To FUSE or Not to FUSE: Performance of User-Space File Systems

fuse & virtio-fs introduction – fuse协议

FUSE协议目前支持49种文件系统操作，以FUSE_WRITE为例，其协议格式定义如下：

```
struct fuse_in_header {
        uint32_t        len;
        uint32_t        opcode;
        uint64_t        unique;
        uint64_t        nodeid;
        uint32_t        uid;
        uint32_t        gid;
        uint32_t        pid;
        uint32_t        padding;
};
```

```
struct fuse_write_in {
        uint64_t        fh;
        uint64_t        offset;
        uint32_t        size;
        uint32_t        write_flags;
        uint64_t        lock_owner;
        uint32_t        flags;
        uint32_t        padding;
};
```

```
/** FUSE page descriptor */
struct fuse_page_desc {
        unsigned int length;
        unsigned int offset;
};

struct fuse_args_pages {
        struct fuse_args args;
        struct page **pages;
        struct fuse_page_desc *descs;
        unsigned int num_pages;
};
```

```
struct fuse_out_header {
        uint32_t        len;
        int32_t         error;
        uint64_t        unique;
};
```

```
struct fuse_write_out {
        uint32_t        size;
        uint32_t        padding;
};
```

pages里存放写操作的数据，可以通过read(2)，splice(2)等传递到用户态文件系统服务进程

fuse & virtio-fs introduction – fuse优势

为什么需要在用户态开发文件系统
优点：

- 开发调试效率高，内核态实现一种新文件系统，且将其推进Linux内核主线，时间跨度会相当长
- 各种高性能特性，比如用户态协议栈，协程，线程池等都可以用来实现用户态文件系统服务
- 故障范围小，用户态文件系统服务进程即使crash，也不会影响操作系统内核
- 分布式文件系统种类非常多，需要通过fuse实现在计算节点提供文件系统服务，目前主流的分布式文件系统CephFS，GlusterFS，S3FS，Alluxio，S3QL等都提供了基于FUSE的文件系统服务

一句话总结，优点非常多，任何人基于libfuse都能快速构建自己的文件系统服务

缺点：唯一的缺点就是性能会较差

- 较多的上下文切换开销，对于小io不友好
- 存在内核态和用户态io数据的拷贝

https://lore.kernel.org/linux-fsdevel/20231228123528.705-1-lege.wang@jaguarmicro.com/T/#u

fuse & virtio-fs introduction - virtio-fs请求队列

virtio-fs本质上就是利用 virtio queue传递fuse文件系统请求，并获取响应。
代码位于 fs/fuse/virtio_fs.c

5.11.2 Virtqueues

0
hiprio
1
notification queue
2…n
request queues

The notification queue only exists if VIRTIO_FS_F_NOTIFICATION is set.

5.11.3 Feature bits

VIRTIO_FS_F_NOTIFICATION (0)
Device has support for FUSE notify messages. The notification queue is virtqueue 1.

hiprio queue 用于处理FUSE_FORGET等高优先级请求
notification queue用于文件系统服务主动向virtio-fs内核模块发送通知
文件读写，目录创建删除等操作都提交到request queues

fuse & virtio-fs introduction - virtio-fs设备加载及使用

在云豹dpu场景，virtio-fs 实现为一种pci设备

5.11.4 Device configuration layout

```
struct virtio_fs_config {
        char tag[36];
        le32 num_request_queues;
        le32 notify_buf_size;
};
```

The *tag* and *num_request_queues* fields are always available. The *notify_buf_size* field is only available when VIRTIO_FS_F_NOTIFICATION is set.

tag
is the name associated with this file system. The tag is encoded in UTF-8 and padded with NUL bytes if shorter than the available space. This field is not NUL-terminated if the encoded bytes take up the entire field.

num_request_queues
is the total number of request virtqueues exposed by the device. Each virtqueue offers identical functionality and there are no ordering guarantees between requests made available on different queues. Use of multiple queues is intended to increase performance.

notify_buf_size
is the minimum number of bytes required for each buffer in the notification queue.

```
static struct virtio_driver virtio_fs_driver = {
        .driver.name            = KBUILD_MODNAME,
        .driver.owner           = THIS_MODULE,
        .id_table               = id_table,
        .feature_table          = feature_table,
        .feature_table_size     = ARRAY_SIZE(feature_table),
        .probe                  = virtio_fs_probe,
        .remove                 = virtio_fs_remove,
#ifdef CONFIG_PM_SLEEP
        .freeze                 = virtio_fs_freeze,
        .restore                = virtio_fs_restore,
#endif
};
```

step1: virtio-fs pci设备加载时，virtio_fs_probe完成virtio-fs设备的probe流程，建立tag到virtio-fs pci的映射

step2: mount -t virtiofs test0 /mnt/test, 其中test0为一个tag,利用该名字找到其对应的virtio-fs pci设备，进行挂载操作。

目录

- fuse & virtio-fs introduction
- virtio-blk vs virtio-fs
- corsica virtio-fs卸载方案
- 性能&收益
- Todo lists

virtio-blk vs virtio-fs

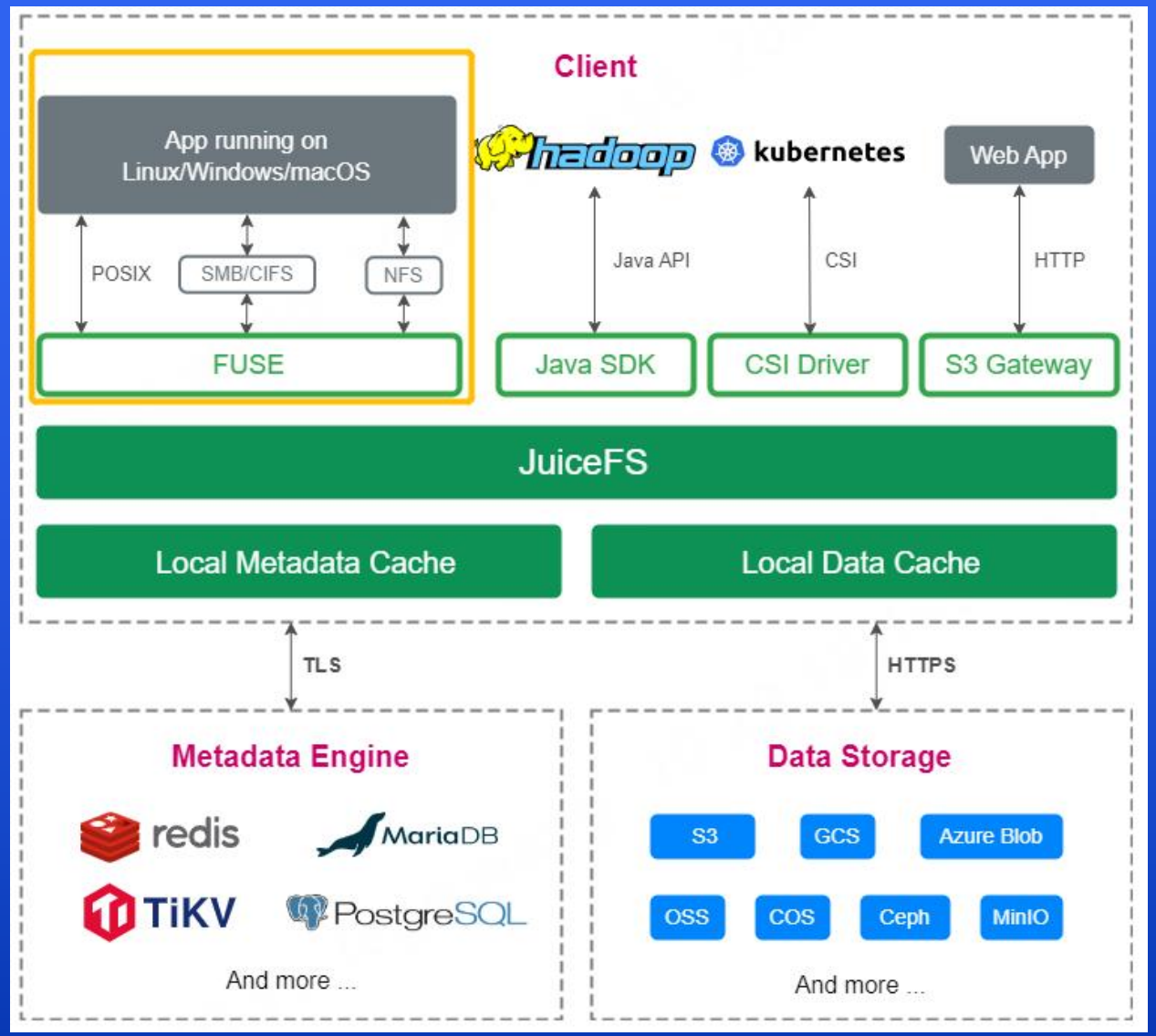
可能的疑惑：

1. virtio-blk通过mkfs.ext4/xfs等工具就可以提供文件系统服务，为什么还需要virtio-fs
2. virtio-fs的性能相比于virtio-blk表现如何

virtio-fs和virtio-blk虽类似，但其是涉及不同的业务场景

- virtio-blk提供块语义的接口，virtio-fs提供文件语义的接口。
- 在云计算场景，难以通过virtio-blk在云服务器之间进行大规模数据共享访问(读写)，不利于AI等计算场景。
- 构建在块设备上的文件系统的容量一般也是有限制的，扩缩容都相对麻烦
- virtio-fs支持所见所得，更贴近业务

virtio-blk vs virtio-fs - fuse在典型分布式文件系统应用



摘自 https://juicefs.com/docs/zh/community/architecture

juicefs支持数千个计算节点同时共享访问同一个文件系统，virtio-blk无法替换此处fuse的位置。

fuse文件系统出现的地方，virtio-fs都可以发挥作用

目录

- fuse & virtio-fs introduction
- virtio-blk vs virtio-fs
- corsica virtio-fs卸载方案
- 性能&收益
- Todo lists

corsica virtio-fs卸载方案 - 现状

工业界与学术界都对 virtio-fs 的卸载进行过相关研究，主要有：

1 intel spdk 团队尝试在 spdk 中支持virtio-fs服务后端

2 学术论文 DPFS: DPU-Powered File System Virtualization 基于nvidia blueField dpu。

上述方案的若干缺陷：

1. intel 的方案基于 spdk BlobFs 组件，BlobFs 组件文件系统能力比较薄弱，只能提供有限的功能
2. DPFS 论文本身很好的说明了卸载文件系统服务到 DPU 侧的优势，但其提出的软件架构基于 nvidia 相关非开源组件，且编程框架基于 polling, 需要对文件系统服务本身做较大的侵入性修改。

nvidia目前正在为bluefield进行virtio-fs特性增强，贡献了virtio-fs内核的多队列功能：
https://github.com/torvalds/linux/commit/529395d2ae6456c556405016ea0c43081fe607f3

corsica virtio-fs卸载方案 - 设计目标

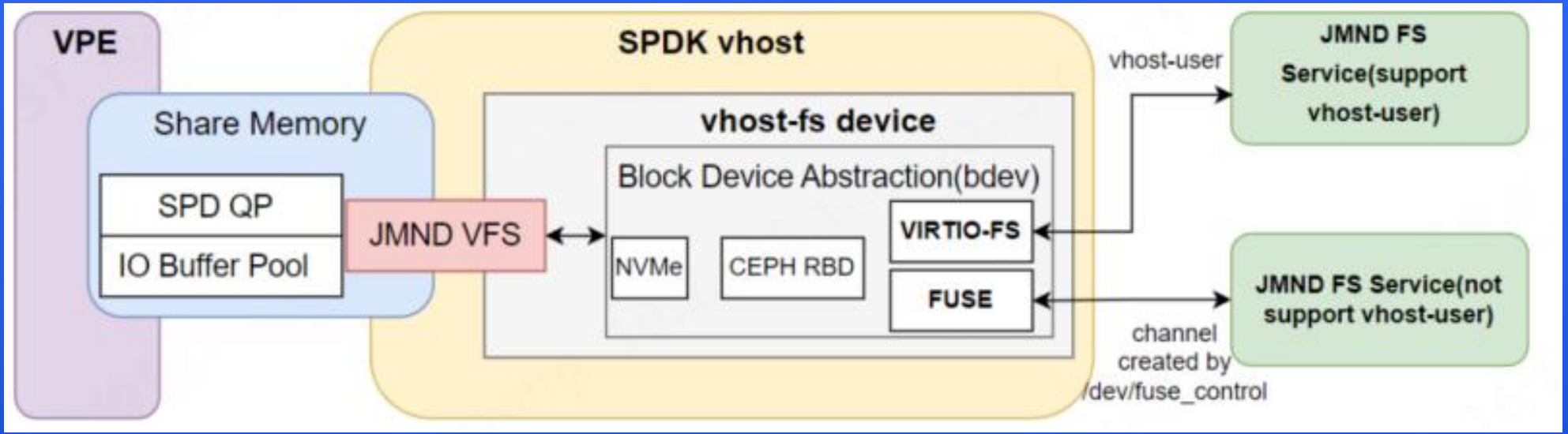
结合业界的实践及我们的思考，提出如下设计目标:

- 基于开源生态的成熟组件，支持所有基于 fuse 的文件系统尽可能无缝卸载到 DPU 侧，这是最重要的设计目标，从而后续的poc验证，部署，运维都会高效。

- 性能需要保证，至少不低于非DPU方案，且能显著降低host侧文件系统服务资源开销

corsica virtio-fs卸载方案 - 架构

整体架构



核心组件有JMND VFS，SPDK virtio-fs bdev, SPDK fuse bdev， JMND FS Service

JMND FS Service按是否支持vhost-user协议分为两类，分别由不同的SPDK bdev引擎对接。

corsica virtio-fs卸载方案 – 核心组件一

- JMND VFS组件

  corsica目前在硬件层面能支持所有virtio设备的卸载，SPDK中的JMND VFS组件通过dma的方式，将host侧fuse请求搬移到dpu侧，将请求提交到SPDK virtiofs相关bdev。

- SPDK virtio-fs bdev

  SPDK virtio-fs bdev 支持vhost-user协议，其作为vhost-user协议的frontend与支持vhost-user协议的用户态文件系统服务交互。目前除virtiofsd外，大多数基于fuse的用户态文件系统服务不支持vhost-user协议，我们实现了libfuse & qemu vhost-user库融合项目，可以支持将基于libfuse的文件系统透明转换成一个vhost-user virtio-fs device backend

corsica virtio-fs卸载方案 – 核心组件二

- SPDK fuse bdev

  host和dpu拥有不同的操作系统, 因此/dev/fuse不可用，我们自研一种进程间传递FUSE报文的高速通道，支持aio，splice等高级特性, SPDK fuse bdev向/dev/dpu_fuse_control字符设备提交请求及获取响应，基于libfuse的用户态文件系统服务通过该字符设备获取请求及写回响应，用户态文件系统服务可以无缝卸载到dpu侧。

- libfuse库增强

  libfuse库的fuse_custom_io特性支持用户态文件系统服务从非/dev/fuse通道获取请求，/dev/dpu_fuse_control字符设备较好地契合该特性，我们对libfuse库的fuse_custom_io特性进行若干增强，已经贡献到社区。

- 用户态文件系统服务

  用户可以自定义其实现， virtiofsd: https://gitlab.com/virtio-fs/virtiofsd作为一种最开始被引入vhost-user virtio-fs device backend，redhat在持续投入，具有极大参考意义，尤其热迁移，线程模型等。

目录

- fuse & virtio-fs introduction
- virtio-blk vs virtio-fs
- corsica virtio-fs卸载方案
- 性能&收益
- Todo lists

性能&收益 - 性能

使能一个sdma通道

iodepth=32 bs=4k 评估IOPS(小IO)

read

iops	bw	lat
388k	1588MB/s	82.34us

write

iops	bw	lat
361k	1481MB/s	88.33us

rw

op	iops	bw	lat
read	34.7k	142MB/s	472us
write	34.6k	142MB/s	449us

iodepth=32 bs=64k 评估IOPS(大IO)

read

iops	bw	lat
91.6k	6000MB/s	349us

write

iops	bw	lat
85.8k	5622MB/s	373us

rw

op	iops	bw	lat
read	17.1k	1118MB/s	954us
write	17.0k	1116MB/s	923us

稳定性

通过xfstests测试集 https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/

性能&收益 - 收益

- 提供**filesystem as service**抽象, 简化host侧运维
- 降低host侧资源开销，提高云厂商物理服务器的资源售卖率
- DPU侧文件系统服务直接走云厂商内部的underlay网络，无overlay网络开销。

目录

- fuse & virtio-fs introduction
- virtio-blk vs virtio-fs
- corsica virtio-fs卸载方案
- 性能&收益
- Todo Lists

Todo Lists

- virtio-fs dax特性dpu支持

  详细信息参见：https://virtio-fs.gitlab.io/design.html

- 零拷贝

  corsica存储栈需要spd ring等硬件将host请求搬移到卡侧进行处理，需要实现卡侧直接访问host内存的方案

- 基于业界典型fuse文件系统服务的卸载验证

  已经完成sshfs等卸载dpu验证，需要补充分布式文件系统的验证

- FUSE协议的跨架构兼容性

  host与dpu不同的字节序，Linux内核文件系统接口在不同cpu架构间的兼容性问题

  https://lore.kernel.org/linux-fsdevel/SI2PR06MB53852C83901A0DDE55624063FFF32@SI2PR06MB5385.apcprd06.prod.outlook.com/

THANKS

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler