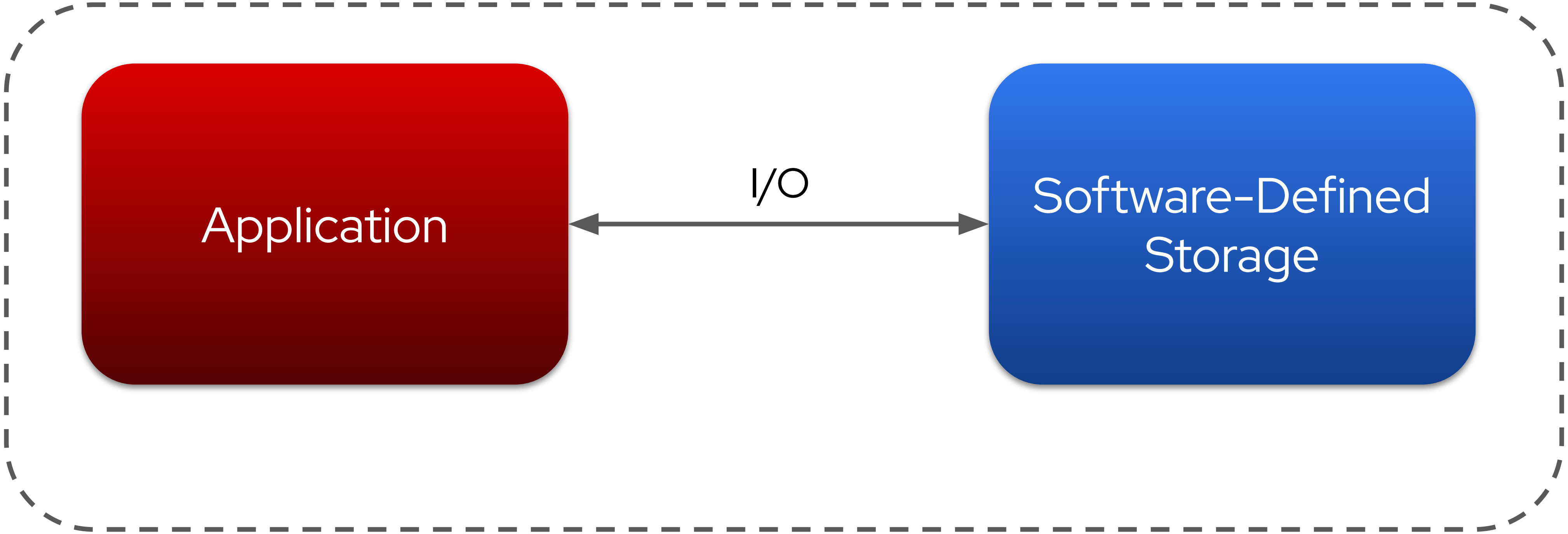
vhost-user-blk

A fast userspace block I/O interface

Stefan Hajnoczi
stefanha@redhat.com

What is vhost-user-blk?



Single node

Software-Defined Storage Models

Block

Fixed-size LBA space

Block-addressable

File

Directory hierarchy

Variable-length files

Byte-addressable

Object

Write once

Read many

Blob storage

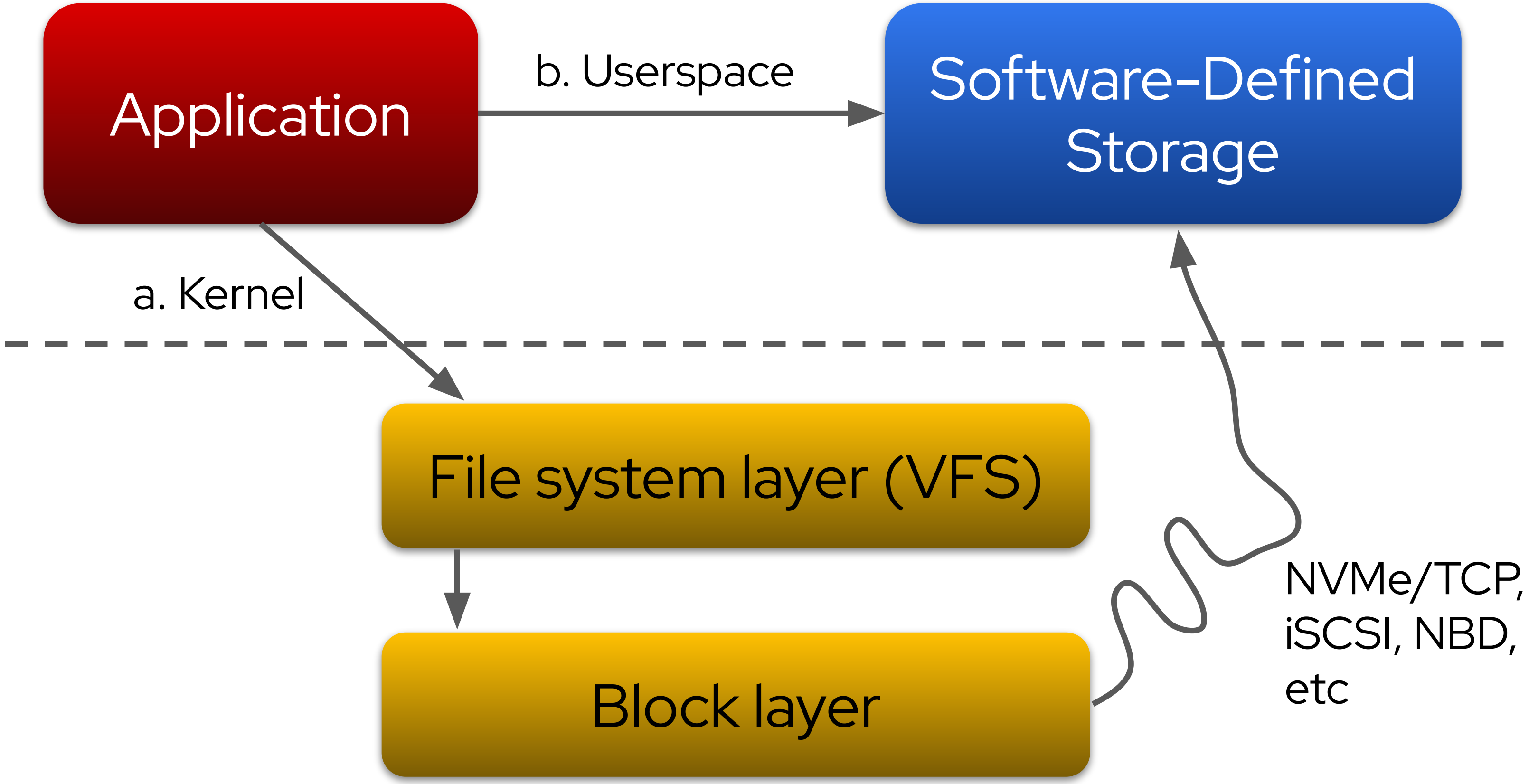
Block I/O Interfaces

Functionality:

- **Core I/O**: Read, write, flush
- **Data management**: Discard, write zeroes
- **Auxiliary**: Get capacity, etc
- **Extended models**: Zoned storage

vhost-user-blk is at a similar level of abstraction as NVMe, SCSI, etc.

Kernel vs Userspace Interfaces



Application

b. Userspace

Software-Defined Storage

a. Kernel

File system layer (VFS)

Block layer

NVMe/TCP, iSCSI, NBD, etc

Userspace Interfaces: Pros and Cons

Fast

- No syscalls necessary in data path

Unprivileged

- No kernel block device involved

Secure

- Removes kernel attack surface

Complex

- Much more involved than read(2)/write(2)

Application integration

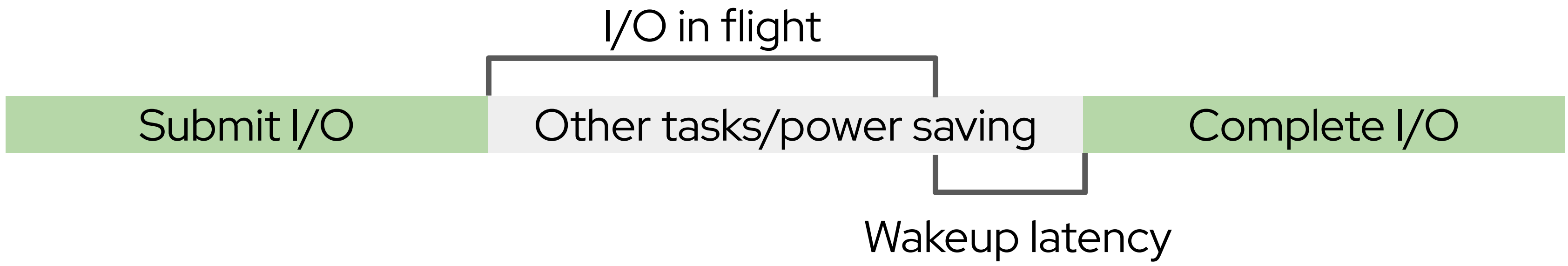
- Existing applications can't use it

Kernel integration

- Separate from kernel VFS/block layer

More later on how to overcome
these things...

Notifications vs Polling



I/O in flight

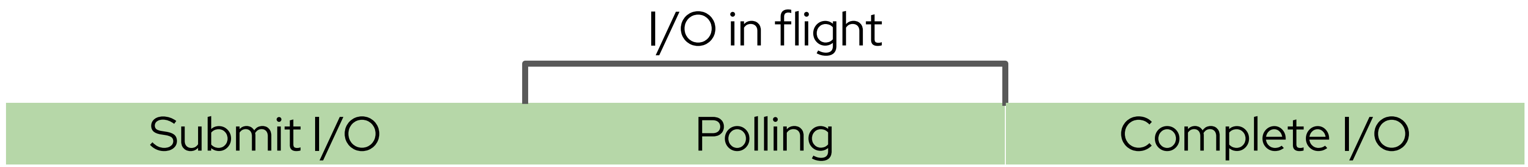
Submit I/O

Other tasks/power saving

Complete I/O

Wakeup latency

Notifications are more power-efficient but have extra latency



I/O in flight

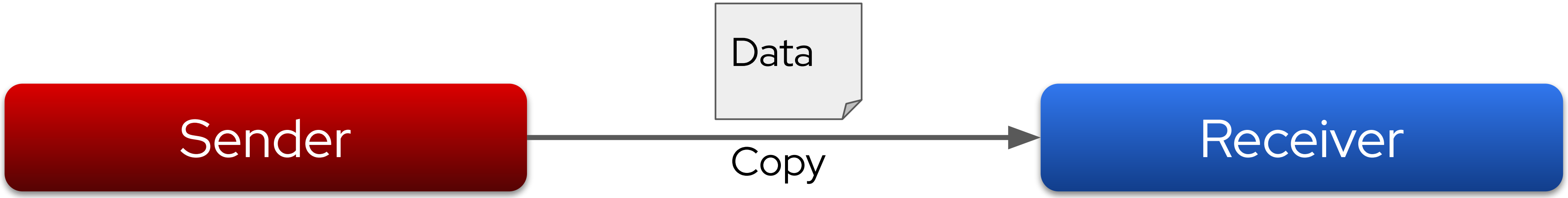
Submit I/O

Polling

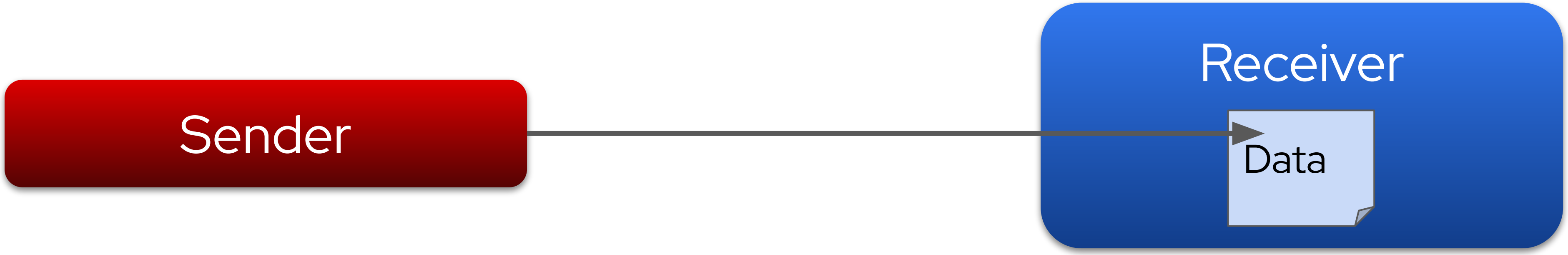
Complete I/O

Polling avoids wakeup latency but hogs the CPU

Message Passing vs Zero Copy



Message passing involves intermediate copies



Zero copy involves access to the final memory, no intermediate copies

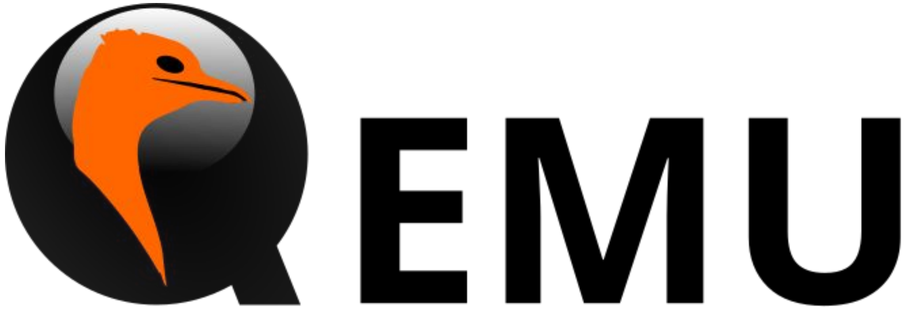
vhost-user-blk

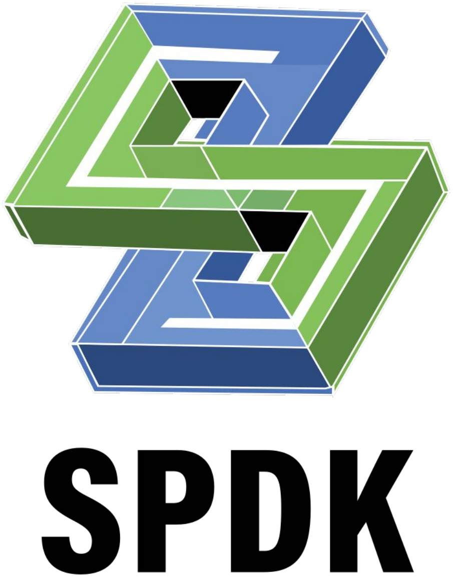
1. Local block I/O interface
2. Userspace
3. Zero-copy (shared memory)
4. Notifications and polling

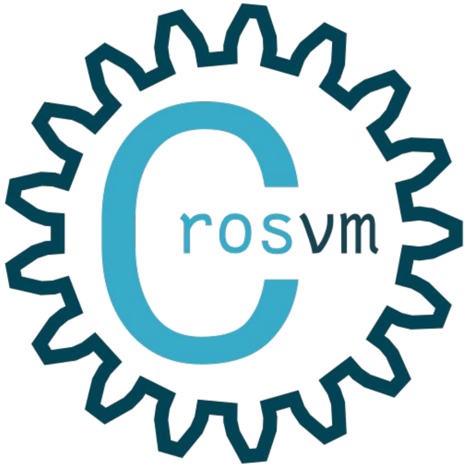
Linux, BSD, and macOS

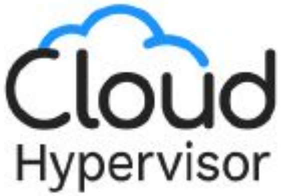
Implementations started in 2017

Where is vhost-user-blk used?







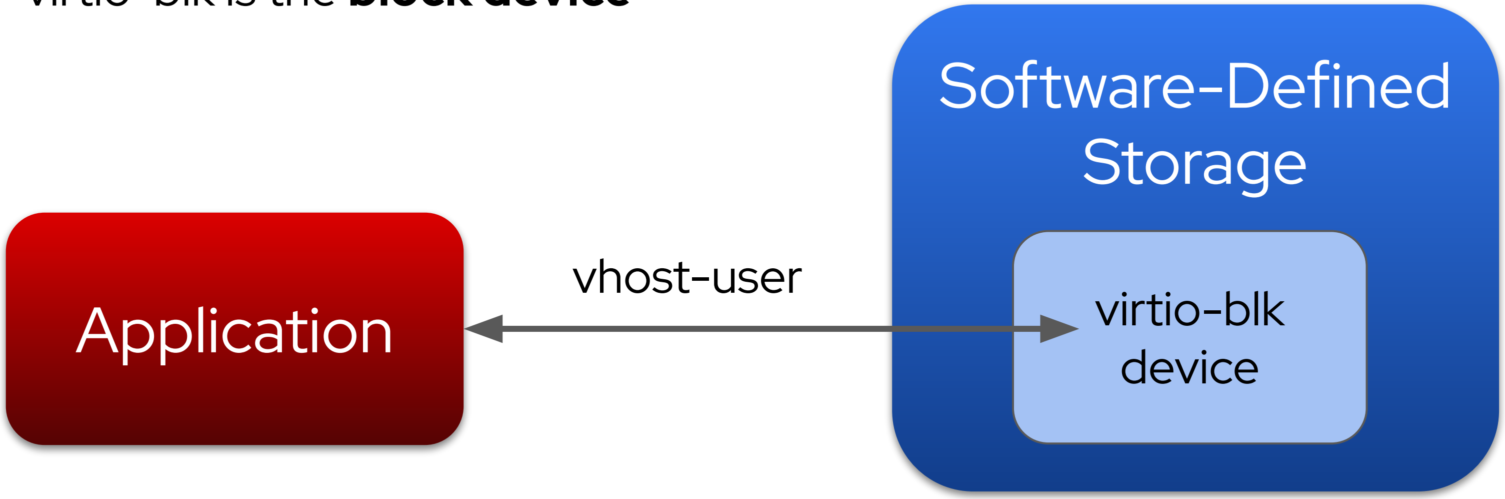


Protocol Overview

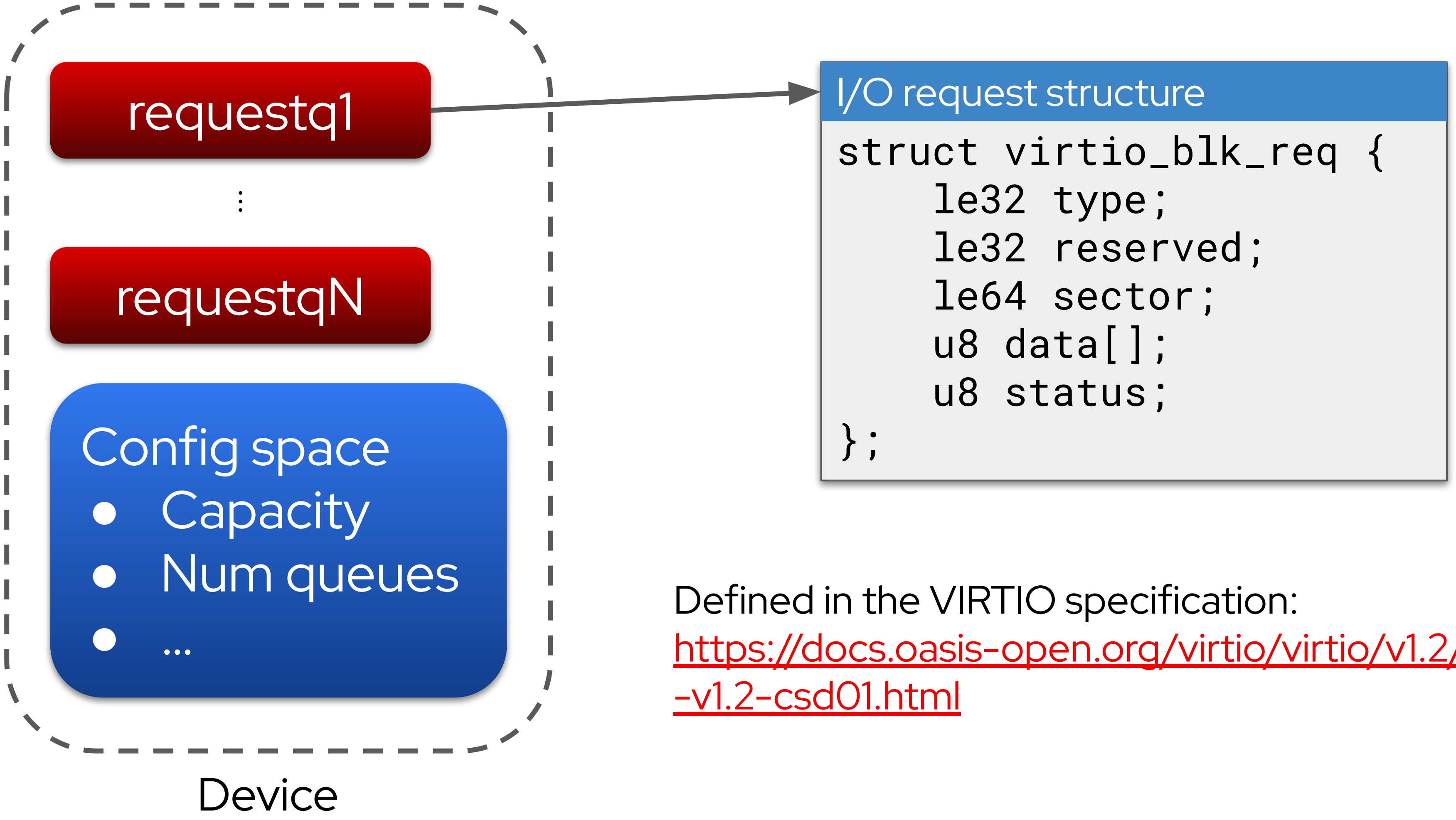
UNIX Domain Socket for **vhost-user** protocol

vhost-user provides access to **virtio-blk** device

virtio-blk is the **block device**



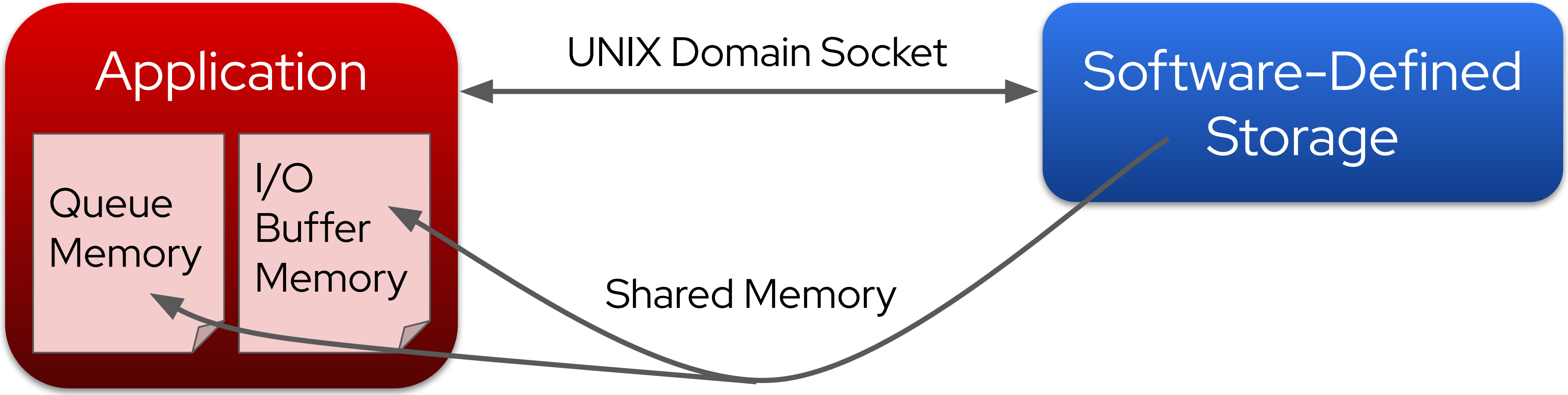
virtio-blk



```
struct virtio_blk_req {
    le32 type;
    le32 reserved;
    le64 sector;
    u8 data[];
    u8 status;
};
```

Defined in the VIRTIO specification:
https://docs.oasis-open.org/virtio/virtio/v1.2/csd01/virtio-v1.2-csd01.html

vhost-user



Protocol for setting up virtqueues for shared memory access

Defined in vhost-user specification:
https://www.qemu.org/docs/master/interop/vhost-user.html

Connecting with libblkio

C & Rust client library

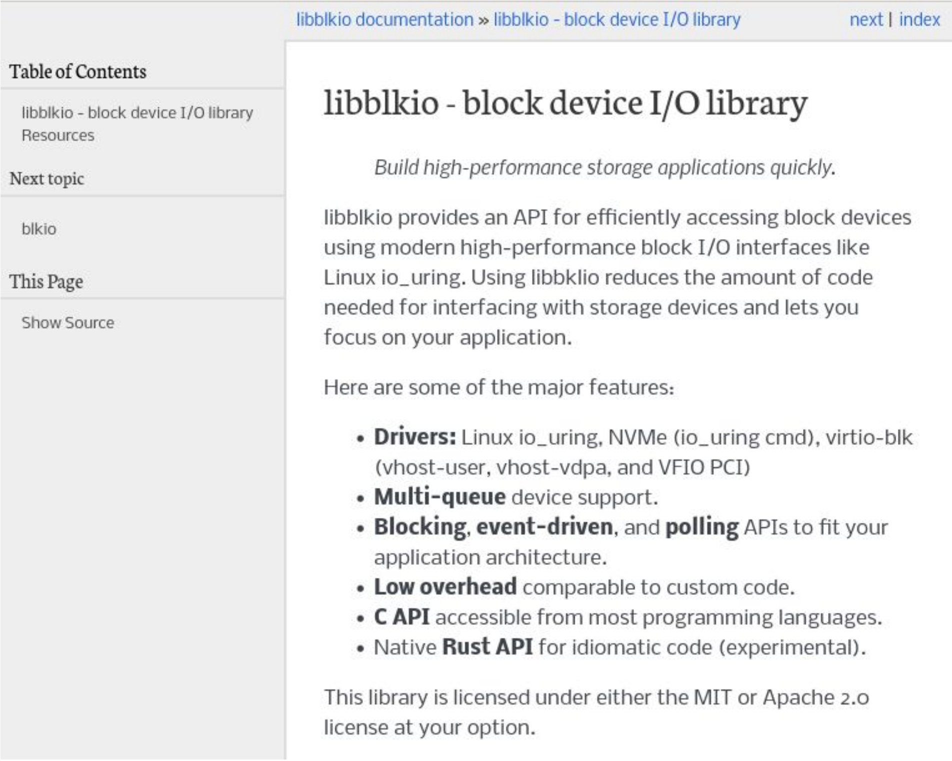
Supports vhost-user-blk and more

Supports blocking, event-driven, and polling I/O

MIT OR Apache-2.0

See the KVM Forum 2022 talk for an overview:

https://www.youtube.com/watch?v=0do1fHPFT8Y



libblkio documentation » libblkio - block device I/O library   next | index

Table of Contents

libblkio - block device I/O library
Resources

Next topic

blkio

This Page

Show Source

libblkio - block device I/O library

Build high-performance storage applications quickly.

libblkio provides an API for efficiently accessing block devices using modern high-performance block I/O interfaces like Linux io_uring. Using libbklio reduces the amount of code needed for interfacing with storage devices and lets you focus on your application.

Here are some of the major features:

- **Drivers:** Linux io_uring, NVMe (io_uring cmd), virtio-blk (vhost-user, vhost-vdpa, and VFIO PCI)
- **Multi-queue** device support.
- **Blocking**, **event-driven**, and **polling** APIs to fit your application architecture.
- **Low overhead** comparable to custom code.
- **C API** accessible from most programming languages.
- Native **Rust API** for idiomatic code (experimental).

This library is licensed under either the MIT or Apache 2.0 license at your option.

https://libblkio.gitlab.io/libblkio/

libblkio C API

Example code without error handling and I/O buffer setup

Setup

```
struct blkio *b;
blkio_create("virtio-blk-vhost-user", &b);
blkio_set_str(b, "path", "vhost-user-blk.sock");
blkio_connect(b);
blkio_start(b);
```

I/O Submission

```
struct blkioq *q = blkio_get_queue(b, 0);
blkioq_read(q, 0x10000, buf, buf_size, NULL, 0);
```

I/O Completion

```
struct blkio_completion c;
ret = blkioq_do_io(q, &c, 1, 1, NULL);
if (ret != 1 || c.ret != 0) ...
```

Developing with qemu-storage-daemon

How do I launch a vhost-user-blk device to test my application?

Exporting test.img at vhost-user-blk.sock

```
$ qemu-storage-daemon \
    -blockdev file,filename=test.img,node-name=file0 \
    -export vhost-user-blk,node-name=file0,\
     addr.type=unix,addr.path=vhost-user-blk.sock,\
     writable=on
```

Implementing a server with SPDK

SPDK has vhost-user-blk support built in:
https://spdk.io/doc/vhost.html

Enable it programmatically or via RPCs:

RPC commands to create a vhost-user-blk device

```
$ scripts/rpc.py bdev_aio_create test.img file0 4096
$ scripts/rpc.py vhost_create_blk_controller \
      --cpumask 0x1 vhost-user-blk.sock file0
```

Implementing a server in C

libvhost-user is a C library that implements vhost-user: https://gitlab.com/qemu-project/qemu/-/tree/master/subprojects/libvhost-user

You need to implement virtio-blk:

- Process I/O requests from the queues
- Set the block device size in Config Space

Example: https://gitlab.com/qemu-project/qemu/-/blob/master/contrib/vhost-user-blk/vhost-user-blk.c

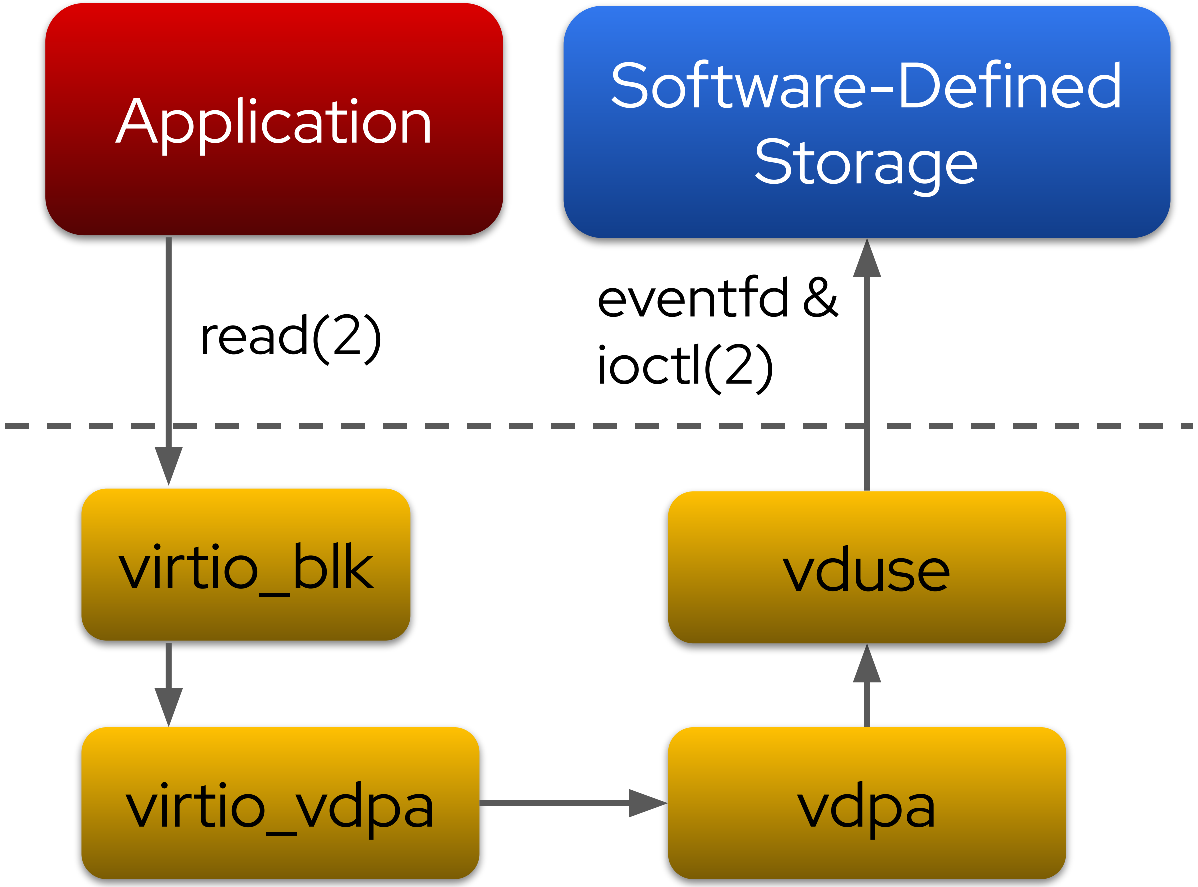
Implementing a server in Rust

Vhost-user-backend is a Rust crate that implements vhost-user: https://github.com/rust-vmm/vhost/tree/main/crates/vhost-user-backend

You need to implement virtio-blk:

- Process I/O requests from the queues
- Set the block device size in Config Space

Exposing Kernel Block Devices with Linux VDUSE



Similar protocol to vhost-user-blk

- Can share code with vhost-user-blk

Uses char device instead of UNIX domain socket

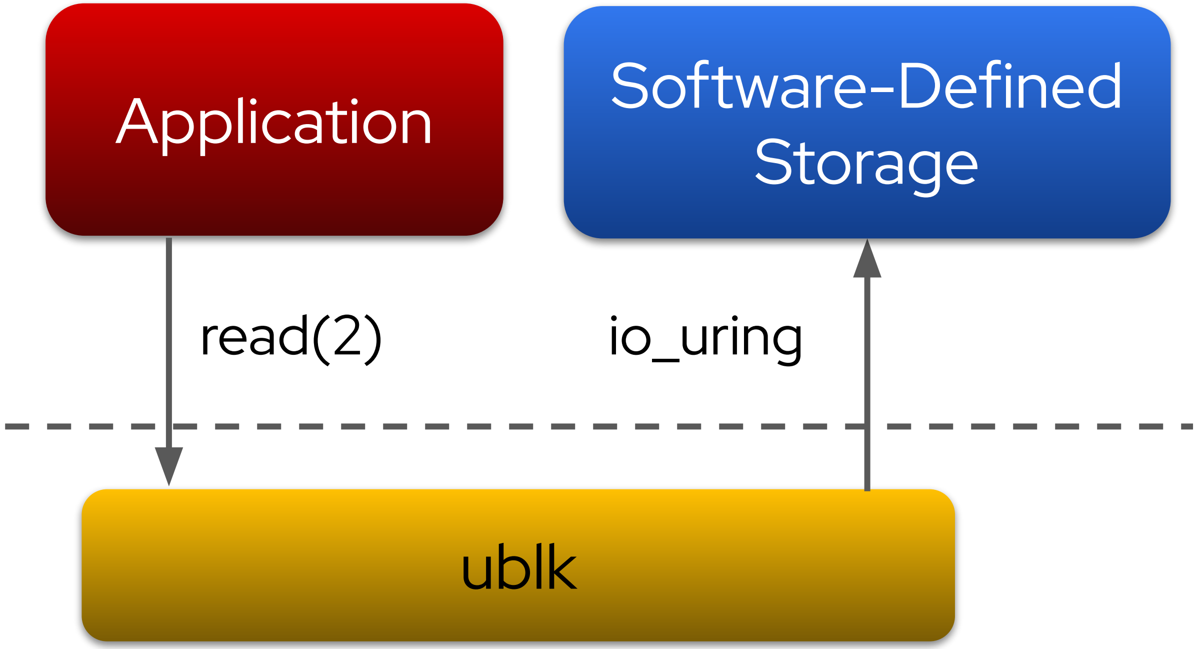
Kernel virtio_blk driver can attach to VDUSE device

- Block device appears as /dev/vda

https://docs.kernel.org/userspace-api/vduse.html

https://gitlab.com/qemu-project/qemu/-/tree/master/subprojects/libvduse

Exposing Kernel Block Devices with ublk



New Linux userspace block I/O interface

No code shared with vhost-user-blk :(

https://docs.kernel.org/block/ublk.html

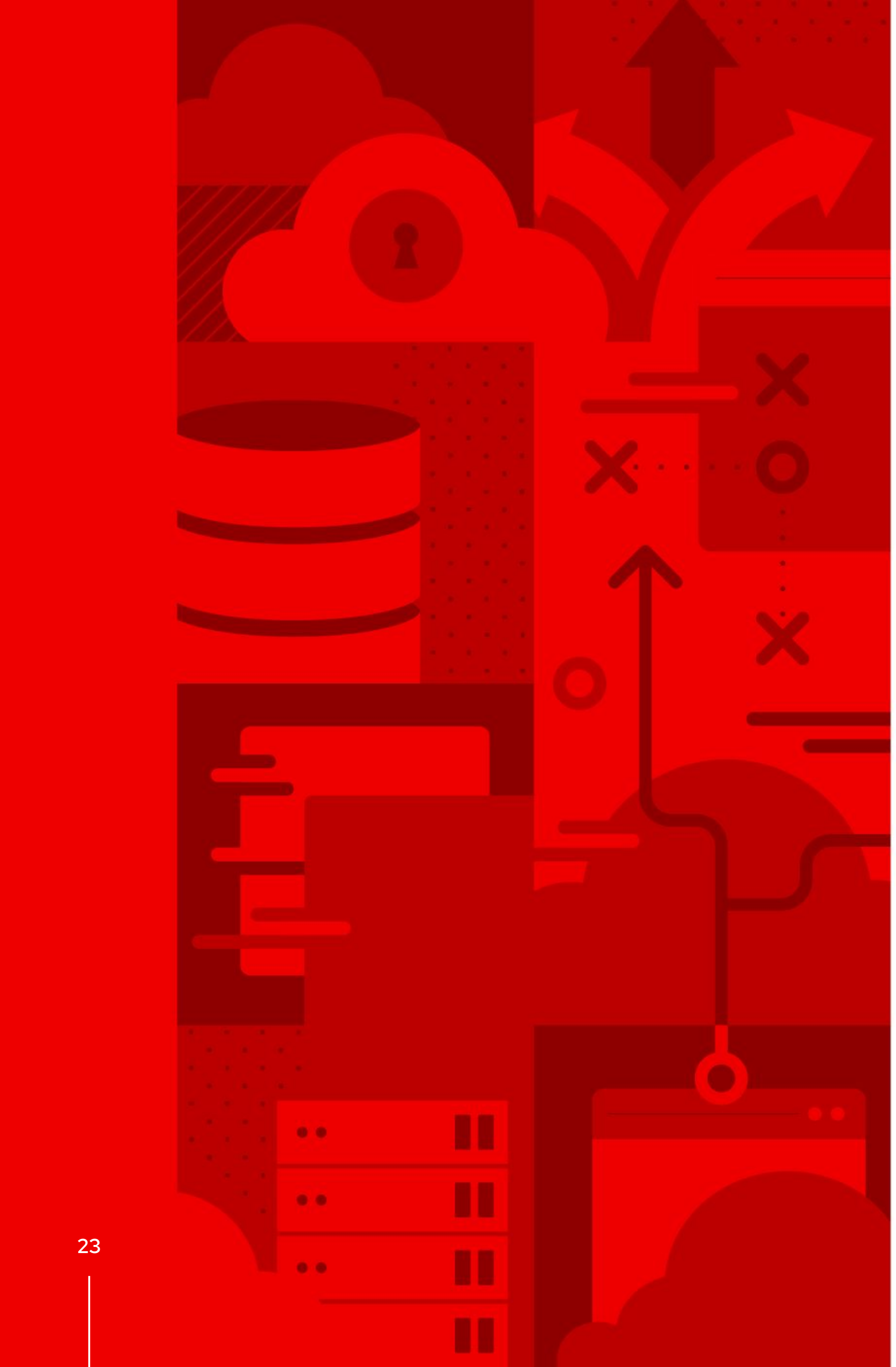
Summary

Need a userspace block I/O interface? (Fast, unprivileged, secure)

Implement vhost-user-blk!

- libblkio for applications (clients)
- libvhost-user, vhost-user-backend, or SPDK for software-defined storage systems (servers)

Open specs, code, and community



Thank you