virtio-fs

A Shared File System for Virtual Machines

Stefan Hajnoczi
stefanha@redhat.com

Red Hat

About me

I work in Red Hat's virtualization team:

virtio-fs virtio-blk tracing

VIRTIO specification open source internships

QEMU Linux

https://vmsplice.net/ "stefanha" on IRC

What is virtio-fs?

Share a host directory with the guest

- ➔ Run container images from host but isolated inside a guest
- ➔ File System as a Service
- ➔ Compile on host, test inside guest
- ➔ Get files into guest at install time
- ➔ Boot guest from directory on host

See KVM Forum talk for "what" and "why":

https://www.youtube.com/watch?v=969sXbNX01U

How to use virtio-fs

"I want to share /var/www with the guest"

Not yet widely available in distros, but the proposed libvirt domain XML looks like this:

```
<filesystem type='mount' accessmode='passthrough'>
  <driver type='virtiofs'/>
  <source dir='/var/www'/>
  <target dir='website'/> <!-- not treated as a path -->
</filesystem>
```

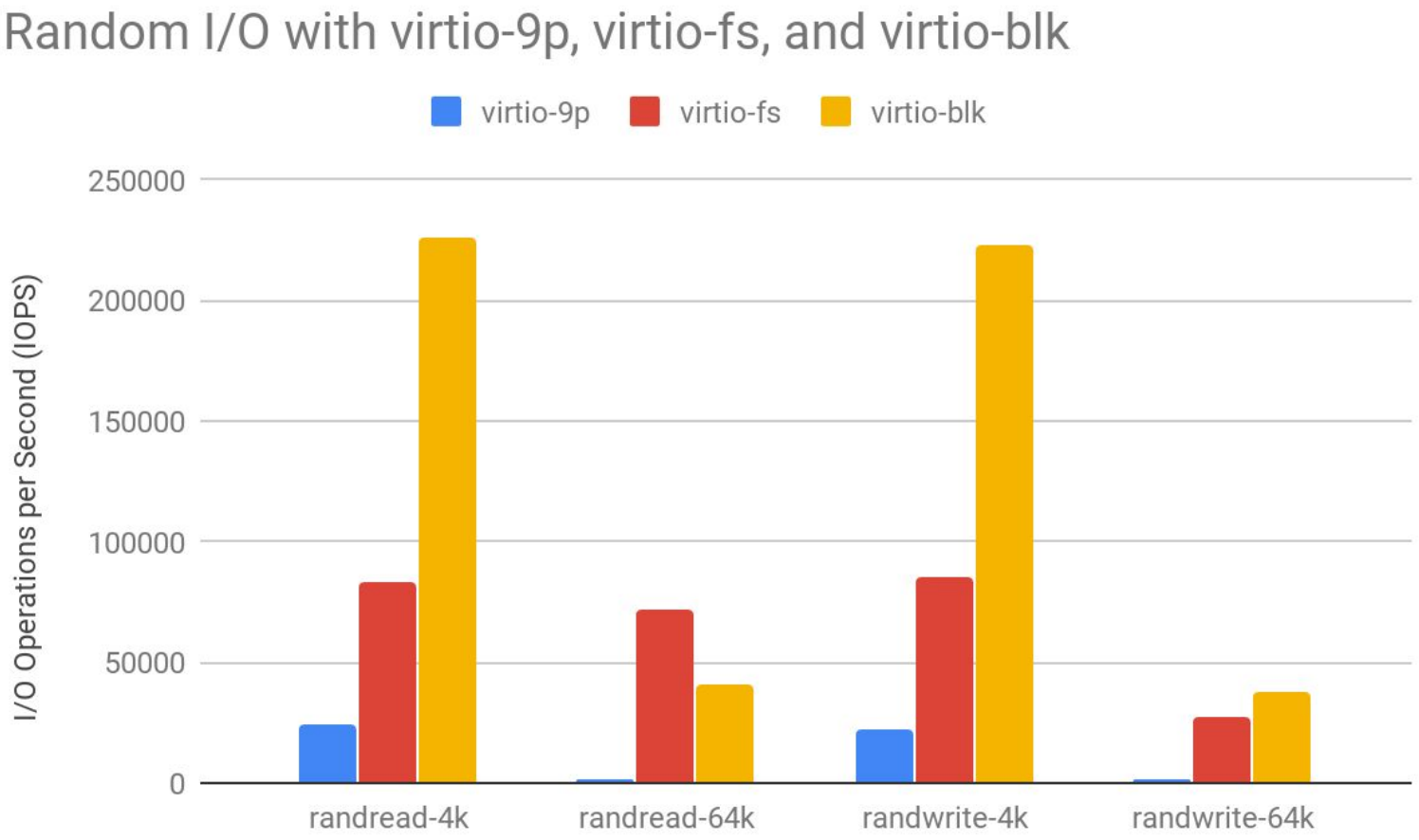
How to use virtio-fs (Part 2)

Mount the directory inside the guest:

```
guest# mount -t virtiofs website /var/www
```

And away you go!

Performance (with a grain of salt)



Out-of-the-box performance on NVMe. Virtio-fs cache=none, no DAX. Linux 5.5.0-rc4 based virtio-fs-dev branch

How do remote file systems work?

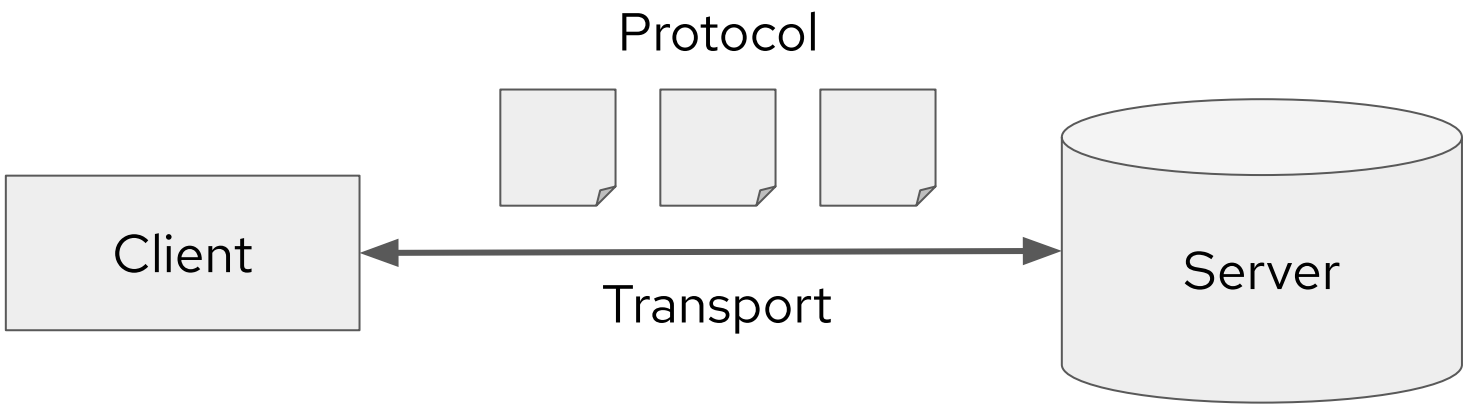
Two ingredients:

1. A **transport** for communication

   TCP/IP, USB, RDMA

2. A **protocol** for file system operations

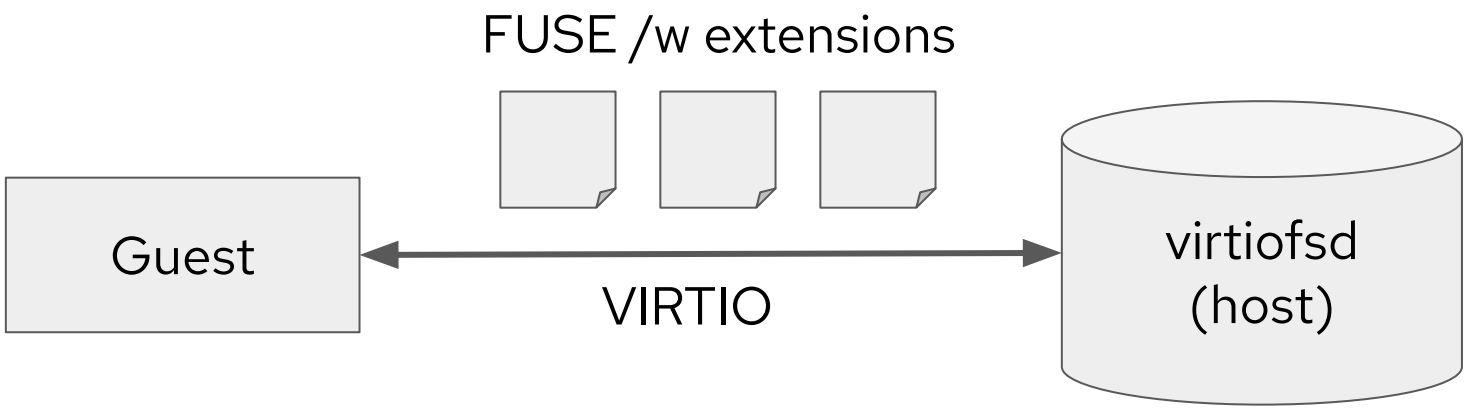
   NFS, CIFS, MTP, FTP



virtio-fs as a remote file system

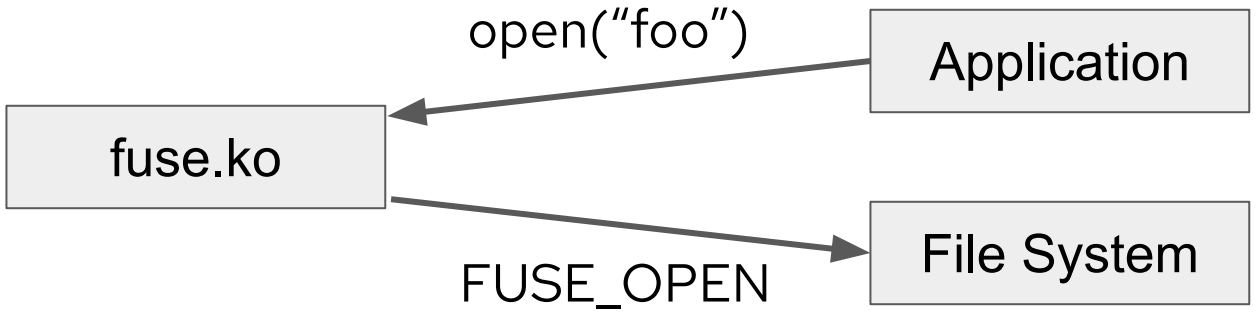
Protocol is based on Linux FUSE

Transport is VIRTIO with shared memory resources



Linux File System in Userspace (FUSE)

Userspace file system interface:



Merged in 2005 and widely available

POSIX semantics + Linux extensions

Extensible protocol

FUSE Protocol

Protocol definitions in <linux/fuse.h>:

```
struct fuse_in_header {
        uint32_t        len;
        uint32_t        opcode;
        uint64_t        unique;
        uint64_t        nodeid;
        ...
};
```

Protocol is undocumented but ABI is stable

Read fuse.ko source to understand protocol

Traditional FUSE

Userspace file system server process

Communication over /dev/fuse character device:

- Server reads next request from /dev/fuse
- Server writes response to /dev/fuse

Server-initiated requests are called notifications and are rare

The virtio-fs Device

Configuration space:

- Tag (mount identifier, e.g. "website")

Virtqueues:

- Requests
- Hiprio (FUSE_INTERRUPT)
- Notifications

Driver places FUSE requests on requests virtqueue

Reading a File

Protocol flow:

1. FUSE_INIT to create session
2. FUSE_LOOKUP(FUSE_ROOT_ID, “foo”) -> nodeid
3. FUSE_OPEN(nodeid, O_RDONLY) -> fh
4. FUSE_READ(fh, offset, &buf, sizeof(buf)) -> nbytes

nodeid is a handle to an inode

fh is a handle to an open file

Bypassing the Guest Page Cache

Can we avoid communication with virtiofsd for every I/O?

Can we avoid copying data to/from host?

Yes! The "dax" mount option will:

- Map regions of files into guest memory space
- Allow guest mmap to directly access data

There is a fixed-size *DAX Window* memory region where host pages are made available to the guest.

Reading a File with DAX

Protocol flow:

1. FUSE_INIT to create session
2. FUSE_LOOKUP(FUSE_ROOT_ID, “foo”) -> nodeid
3. FUSE_OPEN(nodeid, O_RDONLY) -> fh
4. FUSE_SETUPMAPPING(fh, offset, len, addr)
5. Memory access to [*addr*, *addr+len)*

Want Your Own Server?

Virtiofsd passes a directory through to the guest.

But a custom server could:

- Implement its own file system without using file system syscalls on the host
- Directly connect to a distributed storage system
- Export a synthetic file system from the host

See upcoming VIRTIO 1.2 specification for low-level details or use virtiofsd codebase as a starting point.

Thank you

Website: https://virtio-fs.gitlab.io/
IRC: #virtio-fs on chat.freenode.net

Red Hat

virtiofsd Sandboxing

virtiofsd needs privileges to access files with arbitrary uid/gid

What if virtiofsd is compromised by an attacker?

Sandboxing to the rescue:

- Mount namespace only allows access to shared directory (all other mounts are removed!)
- Empty net namespace prevents network connectivity
- PID namespace prevents ptrace of other processes
- seccomp whitelist only allows required syscalls

virtiofsd Security Model

Guests have full uid/gid access to shared directory!

Guests have no access outside shared directory.

Best practices:

- Use dedicated file system for shared directory to prevent inode exhaustion or other Denial-of-Service attacks
- Parent directory of shared directory should have rwx------ permissions to prevent non-owners from accessing untrusted files
- Mount shared directory nosuid,nodev on host