Live Migrating VFIO, vhost-user, and vfio-user Devices

Next steps for out-of-process device migration

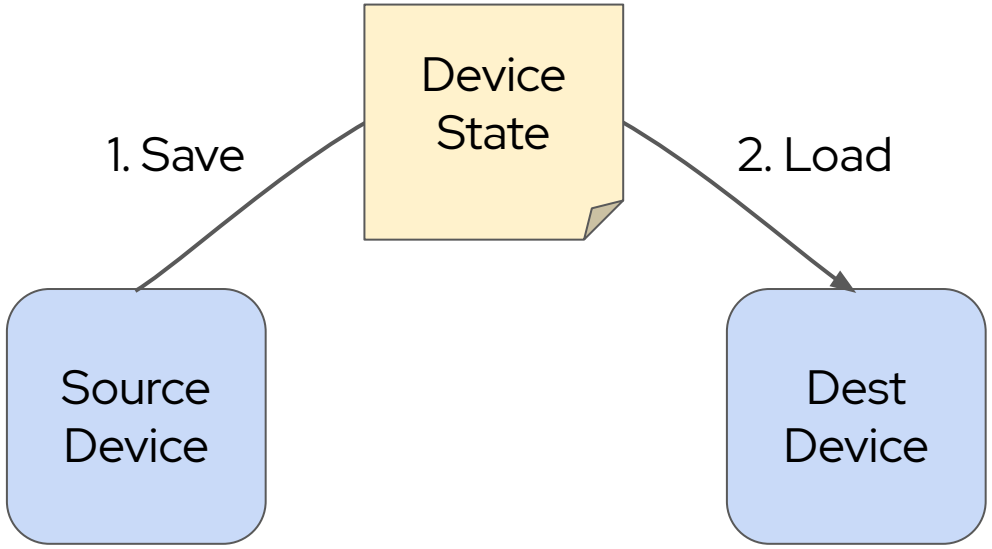
Stefan Hajnoczi
stefanha@redhat.com

Red Hat

Agenda

- How are devices live migrated in QEMU?
- Current approaches:
  - vhost-user
  - VFIO and vfio-user
- Supporting stateful vhost-user device migration
- A proposal for VFIO and vfio-user migration
  - Checking migration compatibility beforehand
  - Migrating between different implementations of a device

How devices are live migrated in QEMU



Device state is saved on the *source* device and loaded into the *destination* device.

Migration is transparent to the guest and ensures continuity from the guest point of view.

Device state representation is the binary layout of a device's state.

What makes up a device's state?

Included:

- Register contents (PCI BAR contents, etc)
- Internal state needed to resume operation (ring buffer reader index, etc)

Excluded:

- Device creation parameters
- Host-side state inaccessible to guest

hw/net/virtio-net.c

```
static const VMStateDescription
vmstate_virtio_net_device = {
  ...
  .fields = (VMStateField[]) {
    VMSTATE_UINT8_ARRAY(mac, VirtIONet, ETH_ALEN),
    VMSTATE_STRUCT_POINTER(vqs, VirtIONet,
      vmstate_virtio_net_queue_tx_waiting,
      VirtIONetQueue),
    VMSTATE_UINT32(mergeable_rx_bufs, VirtIONet),
    VMSTATE_UINT16(status, VirtIONet),
```

Why isn’t everything part of the device state?

Destination QEMU is launched with a full command-line so there is no need to migrate device creation parameters or host-side state.

Destination command-line can be (a bit) different from source command-line!

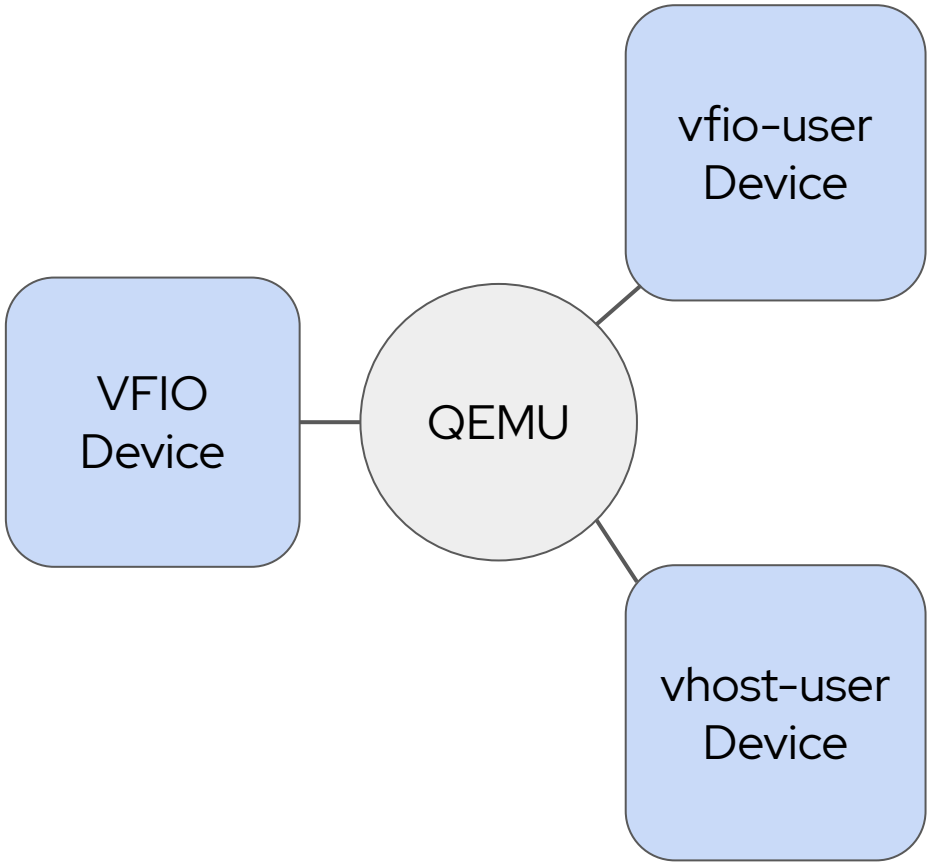
Source

```
$ qemu-system-x86_64 -M pc-q35-6.1,accel=kvm -m 4G ...
```

Destination

```
$ qemu-system-x86_64 -M pc-q35-6.1,accel=kvm -m 4G ... -incoming tcp::1234
```

Challenges migrating out-of-process devices



1. Launching device processes, VFIO devices, etc
2. Integrating into QEMU's live migration workflow
3. Extensibility of the device state representation
4. Migrating between device implementations

vhost-user migration

1. vhost-user device is stopped and QEMU takes over vrings
2. QEMU migrates VIRTIO device using common code
3. vhost-user device is started again on destination
4. Device-specific post-migration steps (e.g. virtio-net VHOST_USER_SEND_RARP)

✅ Easy to migrate between device implementations

✅ Extensible thanks to QEMU's vmstate infrastructure

❌ Cannot migrate stateful devices (yet?), like virtiofs

DBus VMState

Designed for QEMU helper processes but could augment vhost-user live migration for stateful devices

Saves/loads opaque data (up to 1 MB)

Alternative: define device state in VIRTIO or vhost-user specification and integrate save/load into vhost-user protocol

docs/interop/dbus-vmstate.rst

```
Load(in u8[] bytes) method
--------------------------

The method called on destination with the state to restore.

The helper may be initially started in a waiting state (with
an --incoming argument for example), and it may resume on
success.

An error may be returned to the caller.

Save(out u8[] bytes) method
---------------------------

The method called on the source to get the current state to be
migrated. The helper should continue to run normally.

An error may be returned to the caller.
```

VFIO and vfio-user migration

1. Device state is saved by QEMU by reading from VFIO_REGION_TYPE_MIGRATION
2. Device state is loaded by QEMU by writing to VFIO_REGION_TYPE_MIGRATION

✅ Does not require per-device QEMU modifications

❌ Implementor needs to solve extensibility and compatibility issues

❌ It may not be possible to migrate between implementations

Can we check compatibility before migrating?

A cluster scheduler needs to pick a machine capable of being a migration destination.

Options:

1. Optimistically migrate and try another machine if it fails.
   - ➢ *Slow and consumes resources*
2. Use an algorithm to automatically check compatibility.
   - ➢ *Complex*
3. Manually tag machines for cluster scheduler
   - ➢ *Error-prone*

Can we migrate between two implementations of a device?

Compatibility between two devices requires:

- Same device type (e.g. virtio-net-pci)
- Same device creation parameters
- Same device state representation

Source could be a software device, destination could be a hardware device.

Proposal for VFIO and vfio-user migration

Definitions

Device model	Unique domain name + path (ASCII string), e.g. qemu.org/virtio-net-pci
Version	Migration compatibility identifier (ASCII string), e.g. v1

Compatibility check steps

1. Destination supports source device model.
2. Destination supports source version.

Migration steps

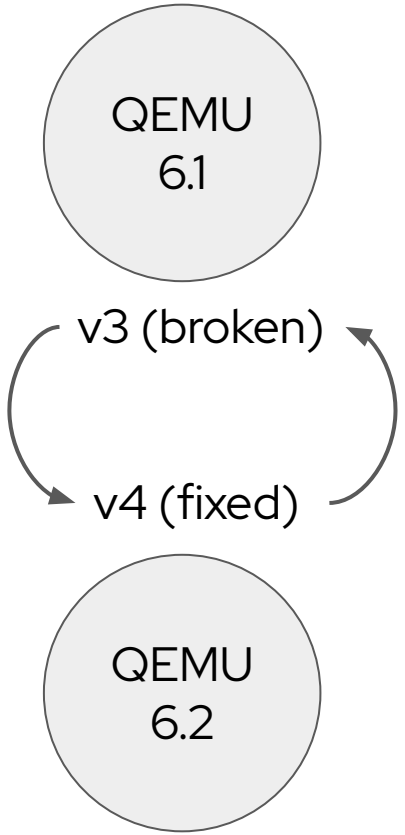
1. Find existing or instantiate new destination device for given device model and version.
2. Save device state on source device.
3. Load device state on destination device.

Upgrading/downgrading versions

Device state representations are sometimes changed to fix bugs.

If the change is migration compatible then a new version string is not necessary. Otherwise:

1. Try to set version string on running device to an older/newer one.
2. The device will refuse if the version change would not be compatible (e.g. changes the guest-visible hardware interface).



How to integrate it

Proposal includes interface for querying and setting device model and version:

- VFIO mdev sysfs attributes
- vfio-user command-line options

This allows management tools to migrate VFIO and vfio-user devices without device-specific knowledge.

Evaluating this proposal

Pros:

- Migration compatibility checking without attempting migration.
- Ability to upgrade/downgrade versions.
- Migration between implementations.
- Easier for implementors than rolling their own device state representations for popular device models.
- Simple concept and not much overhead for implementors and tooling developers.

Disadvantages:

- Does it support everything you need?
- No device creation parameter customization, they are baked into the version to keep things simple (see footnote for more complex alternative).

Migration parameters spec in "[RFC v3] VFIO Migration" (https://lists.gnu.org/archive/html/qemu-devel/2020-11/msg02620.html)

Future directions

vhost-user

- Add support for stateful devices (probably in conjunction with vDPA migration)

VFIO and vfio-user

- Finish VFIO migration proposal discussion & implement it in tooling (libvirt, mdevctl, etc).
- Develop device state representations for popular devices so that multiple implementations can migrate between each other.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat