vDPA support in Linux kernel

Jason Wang
Senior Principal Software Engineer
jasowang@redhat.com
Outline

- Virtio architecture review
- Why vDPA
- vDPA support in Kernel
- Conclusion
- Q&A
Virtio architecture overview

Device Types
- net
- SCSI
- ...

Core device model
- virtqueues
- Feature bits
- Config spaces

Transport
- PCI
- MMIO
- CCW
Software implementation of virtio device

● Several types of virtio devices implemented in software
  – Qemu, vhost-kernel, vhost-user

● Good:
  – Unified device interface for guest
  – Good application usability in guest
  – Live migration support

● Not good:
  – Extra CPU/management cost due to dedicate thread(s)
  – Can’t reach wirespeed due to software overhead
Full hardware implementation of virtio

- A device that is fully compatible with virtio spec
  - control + datapath
  - wirespeed
  - no CPU overhead compared to software dataplane
  - unified API
  - no vendor lock
Issues of hardware virtio implementation

- Current virtio is not designed to be virtualized
  - No support for live migration
- Hard to be integrated with existing hardware
  - Modern hardware is much more complicated
  - Redesign the mature control path is a challenge
- Vendor add-on values requires extensions
- Manageability
  - Lacking features for e.g VF provisioning ...
Why vDPA?

● What is vDPA
  – vhost Data Path Acceleration (originally)
  – virtio Data Path Acceleration

● vDPA is a kind of hardware that has
  – virtio compatible datapath (defined by virtio spec)
  – vendor specific control path (functional equivalent or superset of virtio)
  – vhost features: device state recovery or dirty page tracking (optional)
vDPA device – hardware perspective

- virtqueues
- virtio features
- vhost features
- vendor specific config
Why vDPA

- Advantages of hardware virtio implementation
  - unified datapath
  - wirespeed
- Plus (functional superset of virtio)
  - live migration support
  - vendor specific add-on features(values)
- But still has gaps for E2E delivery if exposing raw vDPA device
  - exposing the complexity and difference to upper layer?
  - integration with existing subsystems or inventing the wheel?
  - manageability, vendor specific management API?
  - heavyweight driver?
vDPA kernel framework

- Bridging the usability and manageability gap
- A framework with the following features is required
  - hiding the complexity and difference
  - presenting a unified device and management API
  - seamless integration with the existing subsystems
  - serving for both userspace drivers and kernel drivers
  - bus/device agnostic
  - lightweight driver
vDPA bus for abstracting hardware

- vDPA bus
  - different vDPA devices and drivers to be attached
  - communication protocol (config operations)
- vDPA device
  - device abstraction provided by vDPA parent device driver
  - vDPA attributes
- vDPA bus driver
  - connect vDPA device to kernel subsystems
  - using config operations to control vDPA device
vDPA parent

- vDPA device for common abstraction:
  - attributes
  - config operations: virtio, interrupt, doorbell, DMA, vhost
- vDPA parent (device) for providing this abstraction
- Parent can be any type, e.g:
  1) parent device driver
  2) intermediate layer on top of another device driver or framework
  3) software device (or proxy)
vhost-vDPA bus driver

- Present a vhost device to vhost subsystem
- Serve userspace drivers
  - VM/Qemu vhost backend
  - DPDK virtio PMD
- Reusing vhost generic uAPI for datapath setting
- New dedicated vhost-vDPA uAPI for a full device abstraction
  - control path commands:
    - config space access
    - status set/get
    - config interrupt
    - ...

vDPA support in Linux Kernel
virtio-vDPA bus driver

- Present a virtio device to virtio bus
- A kernel visible virtio interface via virtio drivers via a new virtual transport (vDPA transport)
- used by various kernel subsystems as if they are virtio device
  - Networking subsystem
  - Storage stack
  - I/O-uring, etc
- bare metal or containerized app
Netlink based management API

- A dedicated vDPA specific netlink protocol for:
  - lifecycle management: create/destroy, enable/disable
  - Provisioning
- A new “vdpa” program will be integrated with iproute2
- All vDPA parent device is required to implement the vDPA netlink protocol
vDPA parents

- Intel IFCVF
  - vDPA is implemented through a dedicated VF
  - parent is a PCI VF device driver
- Mellanox mlx5_vdpa
  - vDPA is implemented through a dedicated VF
  - parent is an intermediate layer on top of mlx5_core module
- VDPA simulator
  - vDPA is implemented through software emulation
- More is being developed
  - ADI or subfunction, endpoint device

vDPA support in Linux Kernel
## Status

<table>
<thead>
<tr>
<th>features</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic function: vDPA core, vDPA bus drivers</td>
<td>merged in Linux</td>
</tr>
<tr>
<td>IFCVF/mlx5e/simulator device</td>
<td>merged in Linux</td>
</tr>
<tr>
<td>basic Qemu support</td>
<td>merged in Qemu</td>
</tr>
<tr>
<td>netlink based management API</td>
<td>WIP</td>
</tr>
<tr>
<td>live migration support</td>
<td>WIP</td>
</tr>
<tr>
<td>control virtqueue support</td>
<td>WIP</td>
</tr>
<tr>
<td>devices other than networking</td>
<td>WIP</td>
</tr>
</tbody>
</table>
## TODO

<table>
<thead>
<tr>
<th>features</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>vDPA device API Documentation</td>
<td>planned</td>
</tr>
<tr>
<td>vhost-vDPA uAPI Documentation</td>
<td>planned</td>
</tr>
<tr>
<td>SVA or vSVA support</td>
<td>planned</td>
</tr>
<tr>
<td>dirty page tracking through hardware</td>
<td>planned</td>
</tr>
<tr>
<td>virtio specification extension</td>
<td>planned</td>
</tr>
</tbody>
</table>
Conclusion

- vDPA device introduced
  - device has virtio datapath with vendor specific control path and vhost features
- vDPA framework in Linux Kernel
  - vDPA bus and device for abstracting and present a unified device interface
  - vDPA bus drivers for connecting vDPA device to various kernel subsystems
- Wire speed virtio with best usability and manageability:
  - no vendor lock
  - live migration (cross vendor/backends)
  - unified management interface
  - mature software stack, virtio ecosystem
Reference

- Steve’s vDPA presentation on KVM Forum 2018

- Redhat blogs for virtio/vDPA

- Virtio specification
  - https://docs.oasis-open.org/virtio/virtio/v1.1/virtio-v1.1.html

- Subscribe to virtio-networking mailing list
  - virtio-networking@redhat.com
vDPA is coming to real life
Please contact to us

- Please contact to us if you want any help:
  - hardware virtio/vDPA implementation
  - driver implementation
  - deployment and integration with management stack
  - feature requirement for implementing your vDPA hardware
  - virtio-networking@redhat.com or jasowang@redhat.com
Thanks