Preserving IOMMU states during kexec reboot

Fam Zheng

System Technology & Engineering team

字节跳动 ByteDance

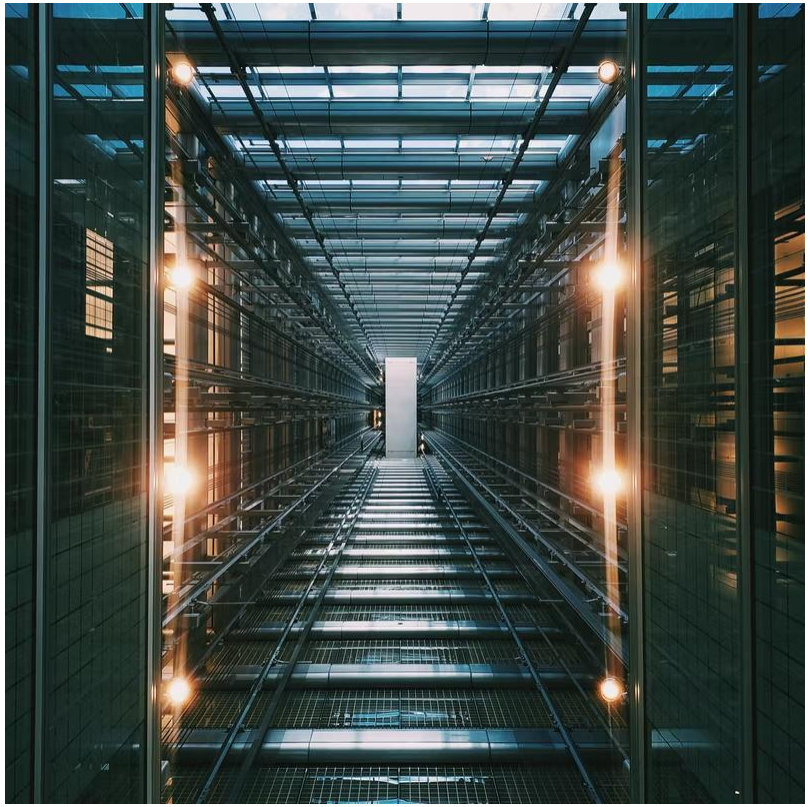
Agenda

- Introduction
  - Motivation
  - Approach comparison
  - VFIO-PCI review
- Live updating vfio-pci
  - Stateful vs stateless
  - Memory considerations
  - Changeset review
- PoC and future plan

Introduction

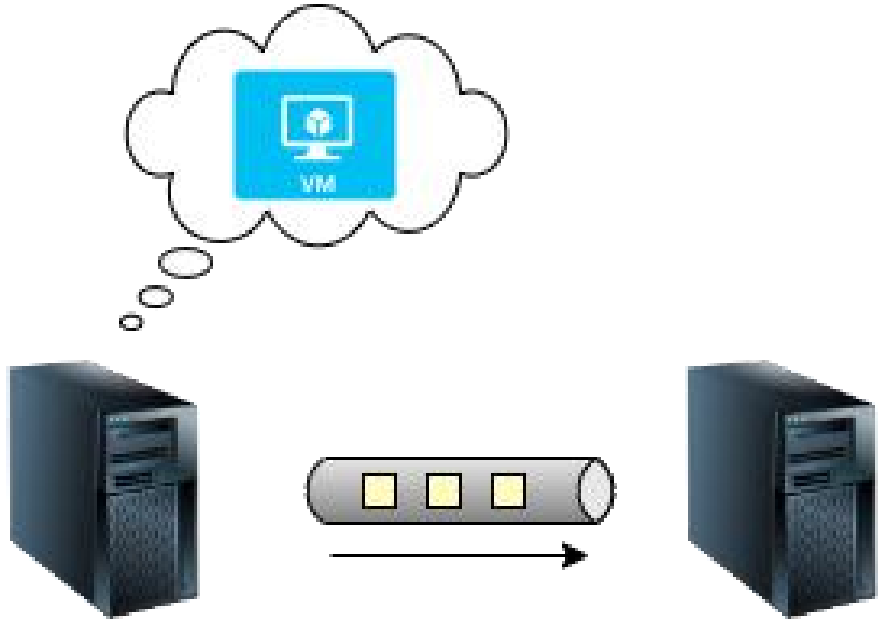
字节跳动 ByteDance

- Long running VMs in cloud should run in a secure, up-to-date hypervisor
- Challenge: how to roll out updates to:
  - Fix security issues
  - Fix functional bugs
  - Bring feature/performance improvements
- Solutions:
  - Live migration
  - Live update



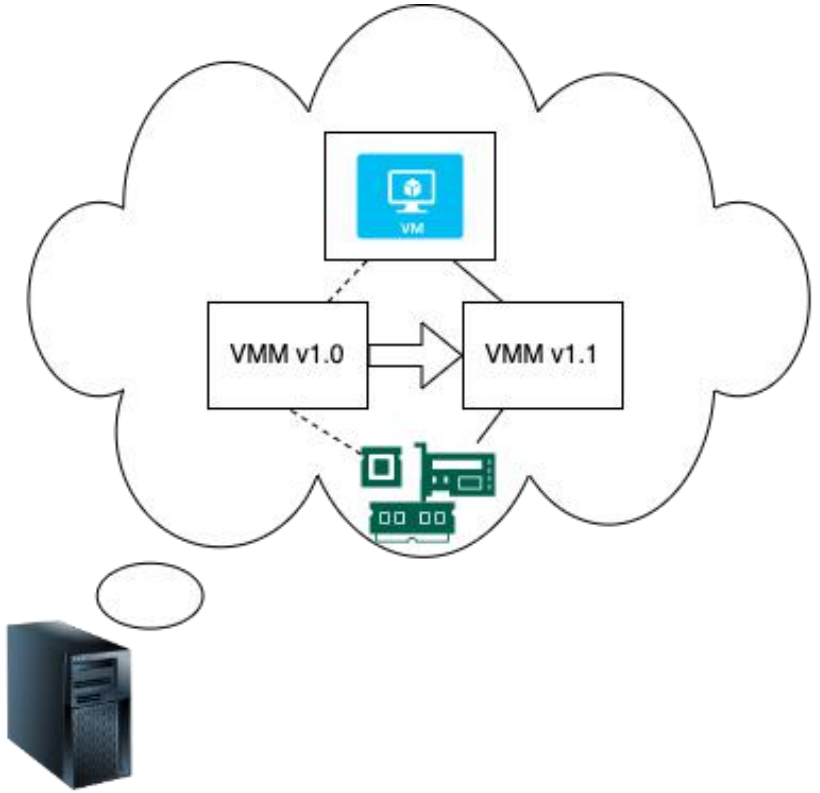
Live-migration

- Live-migration is ...
  - To move the guest from one slot to another
  - Can resolve hw/sw issues while doing this
  - Save / load are usually very heavy operations
  - Challenges:
    - Resource and time
    - Converge
    - Error recovery



Live-update

- A very special case of live-migration:
  - Stay on the same host
  - Carefully avoid excessive data copy
    - Memory is shared
    - Storage data can be accessed easily afterwards
    - Better downtime than live-migration



✓ High bandwidth network for memory copy is not needed

✓ 2x memory is not required

✓ On extra host is not required

- Cannot handle HW issue

Live-update: challenges

字节跳动 ByteDance

- Handing over states and resources can be trickier than live-migration:
  - Memory
  - Network
  - Passthrough devices
- Can host kernel be live updated too w/o compromising above?

Live-update: migrate to file

字节跳动 ByteDance

- Save state
  - QEMU: migrate to <file>
- Update
  - Reboot / exec to start new version
  - QEMU and (optionally) kernel restart
- Load state
  - QEMU: -incoming <file>

vfio-pci quick review

ByteDance 字节跳动

- Abstracting and exposing IOMMU to userspace VMM
- In order to support PCI device direct assigning
- Main tasks
  - Mapping IOVA = GPA --> HPA
    - Passthrough device to get MMIO doorbell notifications
    - DMA to guest ram for accessing hw queue and data payloads
  - Mapping INTX, MSI, MSI-X vectors --> irqfd
    - Passthrough device to interrupt guest driver
- Most setups happen during VM creation:
  - Guest memory layout are stable --> GPAs rarely change
  - Guest pages are “pinned” --> HPAs never change

Live update: cpr reboot with vfio-pci

- Suspend driver with qemu guest agent
- Save state
  - QEMU: migrate to <file>
- Restart / reboot
  - QEMU and (optionally) kernel restart
  - Devices are torn down and reset
- Load state
  - QEMU: -incoming <file>
- Resume guest
  - system_wakeup, to re-initialize guest driver

https://patchew.org/QEMU/1658851843-236870-1-git-send-email-steven.sistare@oracle.com/

Live update: cpr exec with vfio-pci

ByteDance 字节跳动

- Save state
  - VFIO_DMA_UNMAP_FLAG_VADDR
  - QEMU: migrate to <file>
- exec
  - QEMU is replaced with a new binary
  - Various vfio-pci related fds are preserved during exec
    - (Unset FD_CLOEXEC)
- Load state
  - QEMU: -incoming <file>
  - VFIO_DMA_MAP_FLAG_VADDR

Quick recap

字节跳动 ByteDance

Approach	Pro	Con
Live migration	- Well supported - Can deal with hardware issue	- Resource hungry - Higher user experience impact - Converge problem - Failure recovery is tricky
Live update (migrate to file)	- Well supported	- Memory is copied - No vfio-pci support - Very expensive to update kernel
Live update (cpr reboot)	- Update both QEMU and kernel	- Need guest modification to support vfio-pci
Live update (cpr exec)	- Support vfio-pci with unmodified guest	- Cannot update host kernel

Quick recap

字节跳动 ByteDance

Approach	Pro	Con
Live migration	- Well supported - Can deal with hardware issue	- Resource hungry - Higher user experience impact - Converge problem - Failure recovery is tricky
Live update (migrate to file)	- Well supported	- Memory is copied - No vfio-pci support - Very expensive to update kernel
Live update (cpr reboot)	- Update both QEMU and kernel	- Need guest modification to support vfio-pci Really needed?
Live update (cpr exec)	- Support vfio-pci with unmodified guest	- Cannot update host kernel

Zooming into cpr reboot

ByteDance 字节跳动

- If we remove the guest agent, what could go wrong?

- When QEMU exits...
  - vfio-pci teardown
- Old kernel shutdown...
  - device_shutdown
- New kernel start...
  - Device probe & reset
- New QEMU starts...
  - New vfio fd, new DMA and intr mappings

- Inconsistent (broken) state from the guest driver PoV: ERROR!

Fix: Preserving the state

字节跳动 ByteDance

- The way the guest driver wants it:
- The device is not reset in the process
- Config space looks the same as before
- Implicitly, IOMMU, including DMA mappings, MUSTN'T reset
  - Because DMA activities could happen during kernel reboot!

How?

ByteDance 字节跳动

- The key is static physical page allocations, that survive kexec
- Both guest ram pages and host DMAR pages are pinned.
  - But the approaches are different
- INTR is destroyed and re-established, with a spurious notify in the end

Pinning guest ram pages

字节跳动 ByteDance

- memmap=2G!6G

```
memmap=nn[KMG]!ss[KMG]
                [KNL,X86] Mark specific memory as protected.
                Region of memory to be used, from ss to ss+nn.
                The memory region may be marked as e820 type 12 (0xc)
                and is NVDIMM or ADR memory.
```

- ndctl create-namespace -m devdax

- $qemu ... \
  -object memory-backend-file,id=mem,size=2G,mem-path=/dev/dax1.0,share=on,align=2M \
  -numa node,memdev=mem

Pinning IOMMU DMAR tables

ByteDance 字节跳动

- Introducing a static page allocator (KRAM) in kernel
  - memmap=1G:4G（Reserves the region in user e820）
  - Reserving a fixed area:
    kram_get_fixed_page(area, index)
  - Bitmap based allocation:
    kram_alloc_page() / kram_free_page()

Pinning IOMMU DMAR tables

- IOMMU root entries are “fixed” if iommu.kram=1
  - In iommu_alloc_root_entry()...

```
s/alloc_pgtable_page/kram_get_fixed_page
```

- DMAR pages are also in KRAM region so they are “stable” during kexec
  - in alloc_pgtable_page():

```
return kram_alloc_page()
```

vfio-pci integration

- Introduce “raw mode” group fd
  - with VFIO_GROUP_SET_FLAGS
- Skip bus master bit reset during open/close/shutdown etc.
- Same for device reset and config space initialization
- Interrupts vectors are masked before shutdown and unmasked after reboot

Are these enough?

字节跳动 ByteDance

- Unlikely, but let's get going first...

PoC: design and configuration

- Testing setup:
  - QEMU with nested virtualization and virtual IOMMU
  - $qemu -machine q35 -device intel-iommu,intremap=on -cpu host \
    ... -device e1000e,netdev=guestnet
- Patched L1 kernel with vfio-pci live-update support
- Patched QEMU to run L2:
  - Applied CPR patches
  - Use VFIO_GROUP_SET_FLAGS to set device in “raw mode”
  - Added -restore <file> (to be merged into -incoming)

PoC: live-update procedure

字节跳动 ByteDance

- Start L2 with vfio-pci (e1000e)
- Stage new kernel and initrd with kexec_file_load in L1
- QMP cpr-save (tar file written to a DAX blockdev)
- reboot(RB_KEXEC)
- Start L2 again and load state
  - Directly done in a custom /bin/init

PoC: result

- Guest resumes (packets seen on e1000e) within 160ms(*)
- Running on live-updated kernel and QEMU

0 ms: init: save
1 ms: init: cpr-save
1.5 ms: init: qmp: cpr-save [("filename", "/tmp/state/cpr_guest"), ("mode", "reboot")]
18.2 ms: init: qmp done: cpr-save [("filename", "/tmp/state/cpr_guest"), ("mode", "reboot")]
18.3 ms: init: stop
18.3 ms: init: qmp: quit []
20.4 ms: init: qmp done: quit []
21.9 ms: tar: removing leading '/' from member names
21.9 ms: tmp/state/

Total: 159 ms

(*): depends on the size of the test case and hardware configuration

Future plan

字节跳动 ByteDance

- Code clean-up, handle corner cases and errors
- More tests: test on baremetal, in KubeVirt, ...
- AMD and ARM support
- Further downtime reduction during kexec and new kernel boot

Thanks! Questions?

September 2022, KVM Forum, Dublin

Fam Zheng <fam.zheng@bytedance.com>