Supporting Live Update in VFIO

Live Update MC, LPC 2025

David Matlack dmatlack@google.com
Josh Hilke jrhilke@google.com
Vipin Sharma vipinsh@google.com

Goal

- VFIO PCI devices passed into VMs run uninterrupted during Live Update.
    - Devices must not be reset by the host.
    - Devices must be allowed to perform DMA during the Live Update
        - ... to complete ongoing I/O operations initiated by guest
        - ... or handle RX traffic (network device) during Live Update
        - etc.
    - Avoid coordination with guest OS/drivers.
- From guest PoV, Live Update just looks like vCPUs paused for Xs.

Roadmap

Feature	**Version**
Kexec-Handover	v6.16
Live Update Orchestrator and memfd preservation	v6.19
vfio-pci cdev file preservation (without device/HW preservation)	v1
Preserve iommufd and its vfio-pci device attachments	RFC v2
Preserve PCI device state (BARs, Bus Mastering Enabled on parent bridges, etc.)	v2
Keep vfio-pci device in "fully running" state across Live Update.	N/A
Preserve PFs with VFs (SR-IOV)	N/A

vfio-pci cdev file preservation

[PATCH 00/21] vfio/pci: Base support to preserve a VFIO device file across Live Update

- Enables VFIO cdev files to be preserved/retrieved via `/dev/liveupdate`.
  - VFIO resets and restores device just before kexec
    - *This is temporary and will eventually go away!*
- PCI subsystem tracking of which devices are preserved.
  - Used by VFIO to detect userspace trying to open device before retrieving it from LUO.
  - Will be used in future patches to preserve PCI device state (BARs, etc.)
- VFIO selftests to demonstrate the preservation E2E.

Discussion Topic: Interrupts

- If the guest enabled interrupts on the device, the device could signal those interrupts during Live Update when no interrupt handlers are registered.

- Proposed Solution:
  - Require userspace (VMM) disable all interrupts on device prior to kexec.
    - Fail reboot syscall if preserved device still has interrupts enabled.
    - Userspace will need to inject interrupts into guest after Live Update.

- Alternatives:
  - Drop interrupts during Live Update in the IOMMU via IRTE.{P=0, FPD=1}
  - Capture interrupts during Live Update in vCPU Posted Interrupt Descriptors

Discussion Topic: Bus Number Stability

- Bus numbers assigned to preserved PCI devices must be stable to properly translate DMAs through IOMMU during Live Update.

- But the next kernel can assign new bus numbers (`pci=assign-busses`).

- Proposed Solution:
  - Always inherit bus numbers from previous kernel during Live Update (even if `pci=assign-busses` is set).
  - Requires changes to `pci_scan_bridge_extend()`.

Discussion Topic: Device State Identifiers

- PCI subsystem and VFIO need a stable key to match devices with their preserved state across Live Update.

- Proposed Solution:
  - Use PCI segment, bus, device, and function numbers together as this key
    - Segment, device, and function numbers guaranteed stable
    - Kernel must ensure bus numbers are stable for DMA (see previous slide)

- Alternatives:
  - EFI Device Paths (suggested by Lukas Wunner)

Discussion Topic: Driver Binding

- Problem: If a preserved device has a built-in driver, it will bind to the device before vfio-pci.

- Possible Solutions:
  - Fail probe if driver is "incompatible" with the preserved device
    - `pci_driver.name` is most obvious choice, but that would make it ABI [1]
    - Pasha Tatashin proposed using GUIDs embedded in `pci_driver` [2]

  - Require `driver_override` for preserved devices [3]
    - This would embed a specific policy into the kernel, which is brittle
    - No protection from userspace assigning wrong driver and clobbering preserved device

  - Punt PCI driver binding to userspace
    - No protection from userspace assigning wrong driver and clobbering preserved device
    - Which devices? All?

Discussion Topic: pci_saved_state ABI

- VFIO wants to preserve `struct pci_saved_state` for 2 use-cases:
  - `vfio_pci_core_device.pm_save` (restored on transition from D3)
  - `vfio_pci_core_device.pci_saved_state` (restored when userspace done with device)

- Problem: `struct pci_saved_state` is not ABI
  - Layout of individual capabilities can change (pci_cap_saved_data.data[])

- Possible Solution:
  - Change data[] to {register+data}[].
    - The kernel just restores whatever registers are in the `pci_cap_saved_data`, instead of assuming what's in there.
  - Move structs to `include/linux/kho/abi/pci.h`
    - But this violates PCI subsystem desire to keep these structs private.