



VFIO Live Update

Vipin Sharma <vipinsh@google.com>



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

VFIO Live Update

VFIO Live Update (LU) enables a VFIO PCIe device to:



Continue DMA during kernel LU

Maintains uninterrupted memory access during live updates



Avoid BME reset

Prevents Bus Master Enable resets for seamless operation



Avoid device reset

Eliminates device resets during VFIO open



Keep device config same

Preserves device configuration state across transitions



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Current Status (v5)

Timeline & Dependencies

- [v5](#) posted on July 14, 2026
- Built on top of **PCI LU series (v7)**
- PCI LU [v9](#) merged in liveupdate repo on **Sept 28, 2026**

Functionality Scope

Status: Very limited functionality

- VFIO FD preserved using liveupdate IOCTLs
- VFIO FD retrieved **only** via liveupdate IOCTLs
- Opening VFIO cdev file or getting FD via VFIO group ioctl **fails**
- Device resets during kexec as a safety measure (pending full LU)
- Selftests added for verification



Current Status (v5)

Feedback from the VFIO Maintainer (offlist):



VFIO Tree Integration

VFIO Live Update features will go directly through the official VFIO tree.



Experimental Config

Adding `VFIO_PCI_LIVEUPDATE` (Experimental) config doesn't instill confidence.



Multi-Phase Hazard

Intrinsic hazard that effort gets abandoned or superseded in a multi-phased approach.



End-to-End Vision

Need to demonstrate a complete end-to-end solution showcasing the overall goal.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Path to Upstream integration?

End to End VFIO LU solution will need IOMMU LU which is WIP and is dependent on VFIO LU too.

Option 1: VFIO LU for noiommu DMA

- Use IOMMUFD **noiommu** feature for VFIO.
- Use `memmap=xx!yy` to create emulated PMEM. Convert it to devdax and use it for DMA.
- No hardware state of iommu required to be preserved.
- Breaks VFIO dependency on iommu LU.
- No experimental config and a *proper* LU support in VFIO.
- https://github.com/shvipin/linux/tree/vfio/no_iommu_lu
(tested on Intel DSA where transfer continues during LU)



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Path to Upstream integration?

End to End VFIO LU solution will need IOMMU LU which is WIP and is dependent on VFIO LU too.

Option 2: Full LU implementation branch

- Create a separate branch which contains both VFIO and IOMMU LU code.
- Build on top of current proposed series in both of subsystem.
- Propose next series based on patches in the demo branch.
- Update demo branch based on submitted version of patches.
- It show actual LU support without disabling IOMMU on host.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



PCI Saved State Preservation?

When VFIO device is opened it saves the current state of PCI device in **pci_saved_state**. When its last FD is closed it writeback that state.

```
struct pci_cap_saved_data {
    u16 cap_nr;
    bool cap_extended;
    unsigned int size;
    u32 data[];
};

struct pci_saved_state {
    u32 config_space[16];
    struct pci_cap_saved_data cap[];
};
```



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



PCI Saved State Preservation?

● ● ● vfio_pci_core_enable()

```
pci_save_state(pdev);
vdev->pci_saved_state = pci_store_saved_state(pdev);
if (!vdev->pci_saved_state)
    pci_dbg(pdev, "%s: Couldn't store saved state\n",
            __func__);
```

● ● ● vfio_pci_core_disable()

```
/*
 * If we have saved state, restore it. If we can reset
 * the device, even better. Resetting with current state
 * seems better than nothing, but saving and restoring
 * current state without reset is just busy work.
 */
if (pci_load_and_free_saved_state(pdev, &vdev->pci_saved_state)) {
    pci_info(pdev, "%s: Couldn't reload saved state\n",
            __func__);
    if (!vdev->reset_works)
        goto out;
    pci_save_state(pdev);
}
```



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



PCI Saved State Preservation?

Option 1: Save State in KHO

- Add logic in PCIe LU to save `pci_saved_state` in KHO.
- Adds extra maintenance. It can break if new kernel has different saved capability or register.
- There is a [proposal](#) to make its layout more explicit.
- Any change in registers or capability saved will break LU.

Option 2: Skip Saving State

- Don't save `pci_saved_state`.
- Existing VFIO code treats saving and restoring as best effort only.



Live Update finish?

LU provides callbacks `can_finish()` and `finish()` for subsystems to report completion status.

Issue: If a VFIO FD is retrieved from LUO but not bound to IOMMUFD (forgotten or error during bind), what should these APIs do?

Option 1: Block Finish

- Block the LU finish, even if an error prevents the user from binding to iommufd.
- Only a host reboot can make the device reusable.

Option 2: Reset Unopened Device

- If device is not opened (binding to iommufd), reset the device automatically.
- Allows recovery of VFIO LU to a sane state without requiring a reboot.



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Future series?

After VFIO noiommu DMA is merged, which one should be next?

Option 1

1. VFIO **noiommu** SRIOV preservation support
2. VFIO iommu DMA support

Option 2

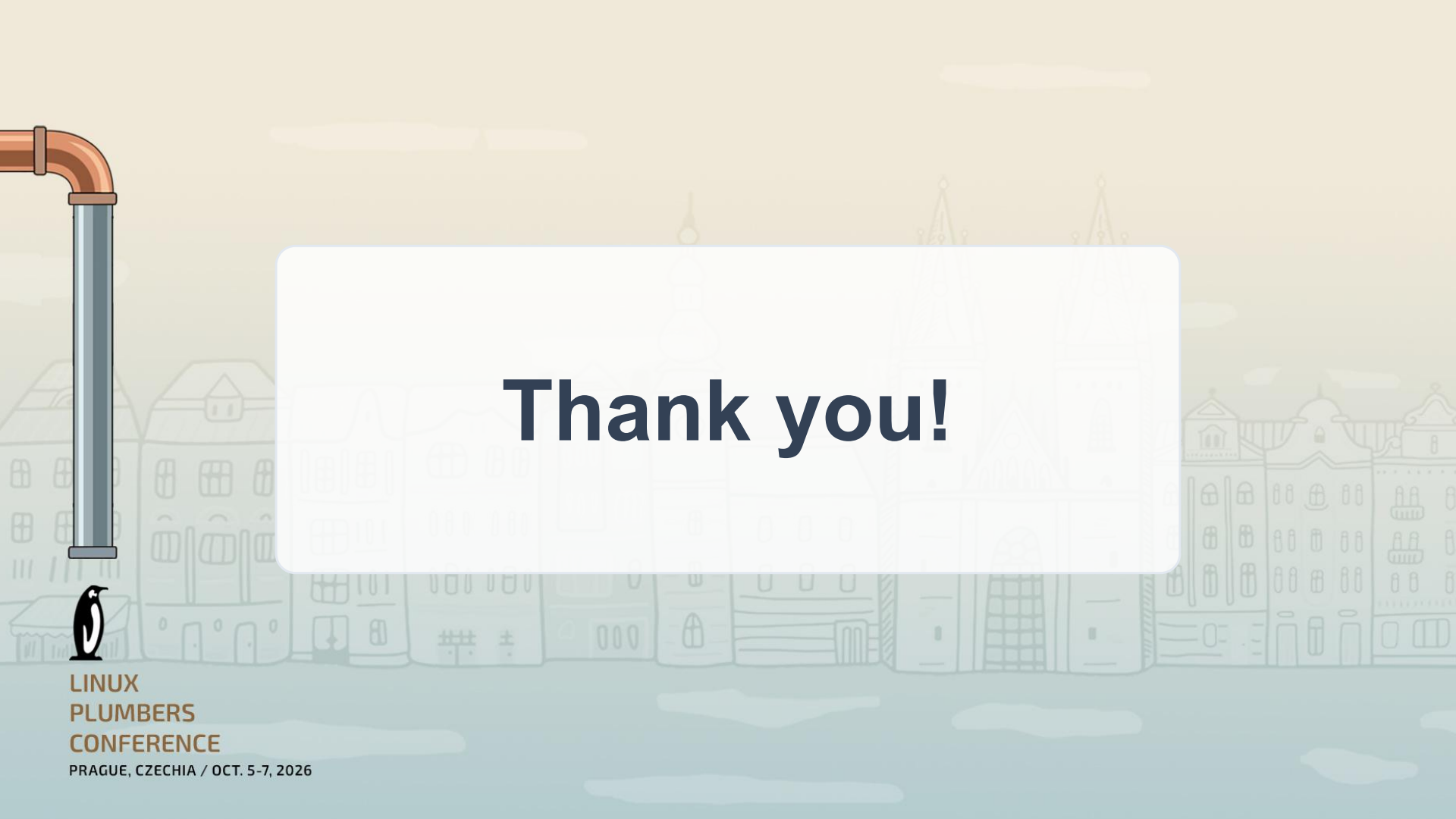
1. VFIO iommu DMA LU support
2. VFIO iommu LU support for SRIOV



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026





Thank you!



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026