Google

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

Google

PCI Subsystem Live-Update

Live Update MC

Chris Li <chriscli@google.com>

東京 2025 Linux Plumbers Conference

TOKYO, JAPAN / DEC. 11-13, 2025

PCI subsystem

Why?

- PCI subsystem is the middle layer between LUO and device drivers.
- BDF must not change cross kexec due to DMA.
- Skip clearing bus master on shutdown.
- After kexec, need initialize the device different (have state)

What?

- Use KHO to preserve PCI device state.

How?

- Build live update device list base on the device dependencies.
- Walk the PCI bus and devices tree to collect liveupdate devices.

PCI device and driver

struct device

- base object for the device tree.
- contain pointer to the device_driver.

struct device_driver

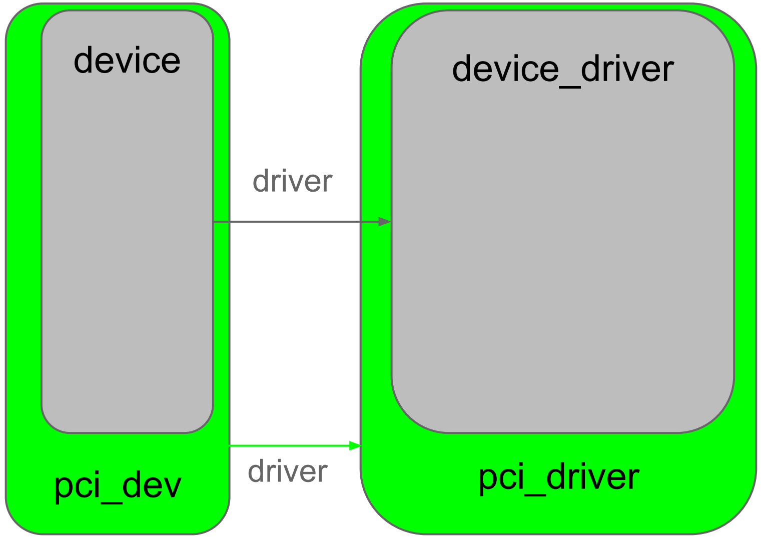
- Pointed by struct device.

struct pci_dev

- Embed struct device object.
- Has pointer to pci_driver.

struct pci_driver

- Embed struct device_driver.



PCI liveupdate device dependency

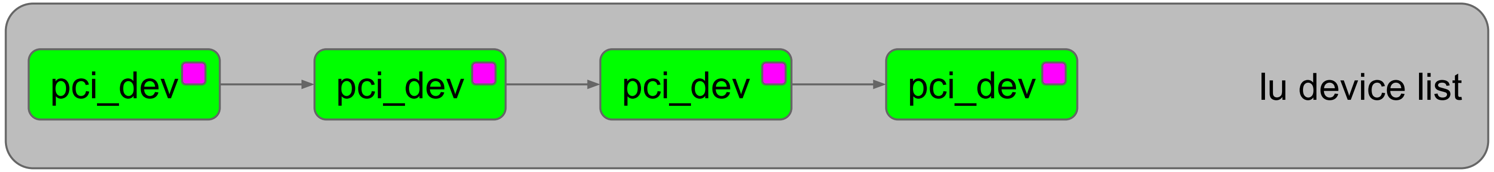
PCI device depends on parent bridge device

- The parent bridge must preserve bus master bit to allow children PCI device continue DMA.

VF device depends on PF device

- The PF device create the VF device.
- The number of the VF device need to be preserved and recreated after liveupdate kexec.

PCI liveupdate device list



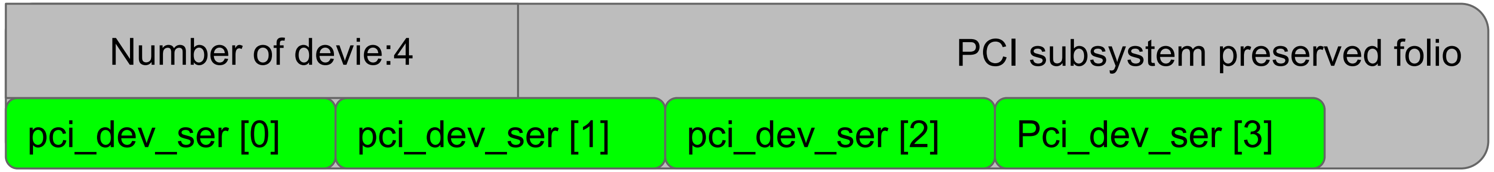
PCI device liveupdate struct

- The structure is add to the struct pci_device to for PCI liveupdate device state.

The PCI liveupdate device list

- A new list out side of the PCI device tree.
- Build the list by walking the PCI host bridge downwards.
- The list is preserved cross kexec.
- After kexec the device initialization avoid fresh init on preserved device.

PCI preserved devices



- The new kernel need to find out which device is participating liveupdate.
  - Avoid auto probe the driver
  - Initialized the device differently (without clearing the device state)
- Each PCI device has some state fields in pci_dev struct to track. (BDF, PF VF)
- Save number of devices and array of pci_dev_ser.

Discussion: PCI device Init Multi-Personality

Context: PCI device initialization will behave differently if device has preserved state or not.

Problem: The current initialization is greedy, enable any capability if supported. This pattern is sprinkering all over the PCI code base. Change is invasive hard to maintain.

Proposed solution: Change to the PCI initialization to separate the scan vs init. Make it three step.
1) scan what is available.
2) choose what to init.
3) perform the actual init.

Discussion: PF init and recreating VF

Problem: The PF is responsible to create the VF after liveupdate. Some PF driver has extra side effect when creating VF. Who is triggering the creation of the VF.

Possible solution:

- Trigger from PCI layer. Will miss the PF side effects.
- Wait for user space to trigger the VF creation after PF driver is loaded. Longer blackout window.

Q&A

Google

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025