Let’s Hot plug:

By uevent mechanism in DPDK

Jeff guo
Intel
DPDK Summit User space - Dublin- 2017
Agenda

- Hot plug overview
- what we have & why uevent?
- Uevent mechanism introduction
- Uevent in virtualization
- Open and plan
- Q & A
Hotplug is a technology, which lets plug in a devices when system is running and use them immediately. While lets unplug a device but not affect the system running.

- HW support(etc. new IA platform), OS support(etc. linux), driver support(etc. OFED)
- Kernel >= linux 2.6, pciehp, port service like
- Management: BIOS -> ACPI.
- Hot-insertion and hot-removal.
- Non surprise hot plug and surprise hot plug.
Hot plug user case

- Load balance
- Reduce power consumption
- Handle hardware error (fail over or fail safe)
- Live migration

For 24/7 availability, don’t take it down for any reason!
what we have.

- General Hot plug API
  - hot plug add / remove,
  - dev_attach / dev_detach,
  - Port plug in & out

- Fail-safe driver
  - like an app helper,
  - Manage sub device and process hot plug event,
  - dynamic switch fail device to safe device.
Currently, device plug & play by plan, it need stop/close port before detach, it would be mass in cloud. And when attach port, need app knowledge the pci device id.

Hot plug event are diversity in drivers, not all uio driver exposure hot plug event, need a general event from bus/device layer.

Uevent is easy to use and management.

- Netlink socket, kobject, asynchronous, sysfs, kernel space --> user space.
- Abundant device status, like add/remove/change/online/offline.
Each component each scope, hot plug belong to device, might be better to offload it from app and driver to the bus/device layer of the eal core lib.

- **pci/vdev, ...**
  - **bus**
    - scan / probe
- **kobject**
  - **device**
    - attach/detach
    - Uevent monitor
- **Igb_uio/vfio/ uio_generic/ other**
  - **driver**
    - bind
    - Initial Operation

Linux Kernel is useful, just use it.
uevent monitor:

- An new epolling, user register interesting event when start.
- A device_state machine in structure of rte_device.

**PARSED/ PROBED / FAULT**

- dev_event_type enumerate and uevent structure in a new file eal_dev.h. BSD not support uevent.

uev_monitor_enable / uev_receive / uev_parse / uev_process/
dev_monitor_start / dev_monitor_stop
Add below API in rte eal device for common

- `rte_eal_dev_monitor_enable`
- `rte_dev_callback_register / rte_dev_callback_unregister`
- `_rte_dev_callback_process`
- `rte_dev_bind_driver`
Failure handler:

- add remap_device in bus layer, to remap the device resource to be “safe” before device detach.
- Add dev_bind_driver in device layer, to auto bind driver before device attach.
- Add find_device_by_name in bus layer, to find device in the device list of bus by the device name
Uevent in virtualization

- Uevent support vfio, each vdev have its own kobject and uevent, it directly process vfio uevent when pf hot plug.
- Live migration, share memory (NFS) or block migration, detect the switching nic across the platform by uevent.
- Uevent for virtio and SRIOV??
Plan and Open...

- Make the API upstream, to public it for developer usage.
- Hot plug API + uevent + failsafe driver, integration and verification.
- Performance(hot plug action speed and packet loss) and robots.
- Co-work with community contributor, fix the gap with pci bus rework.

http://dpdk.org/dev/patchwork/patch/28949/
http://dpdk.org/dev/patchwork/patch/28950/
Questions?

Jeff Guo
Jia.guo@intel.com