PCIe Bandwidth Controller

Ilpo Järvinen

LPC 2024 VFIO/IOMMU/PCI MC

PCIe Bandwidth Controller (bwctrl)

- Controls / throttles PCIe Link Speed
  - Primary motivation: thermal reasons
- Adds internal API to change PCIe Link Speed
- For userspace control, adds a thermal cooling device
  - Provides cur_state under sysfs
  - Controlled by userspace tools such as thermald (out of scope)
- With the advent of PCIe6, Link Width control might be added to bwctrl (future work)

PCIe Link Speed Management

- Link Bandwidth Management Status (LBMS) & Link Autonomous Bandwidth Status (LABS)
  - Asserted when Link Speed (or Width) changes
  - Link Bandwidth Management Interrupt Enable
    - Using interrupt $\Rightarrow$ bwctrl has to be a service driver
  - Allows keeping current Link Speed in struct pci_bus up-to-date
- Supported Link Speeds Vector Link Capabilities 2
- Target Link Speed in Link Control 2

Bandwidth Control Procedure

- Calculate the intersection of supported speeds (Root Port & Endpoint) and pick a speed no higher than the desired speed
  - Handles also gaps in Supported Link Speeds Vectors
    - PCIe6 spec requires no gaps but recommends in an Implementation Note OS to prepare for the possibility
- Adjust Target Link Speed
- Retrain the Link (changes only effective after retraining)
- Observe the results

Bandwidth Controller Integration Challenges

- LBMS already used by the Target Speed quirk
- Bwctrl adds an in-kernel API for setting Target Speed
  - Disabling bwctrl?
- Link Control and Link Control 2 RMW ops must be protected (solved)
  - Heads up: Use RMW capability ops (set/clear) that now take a lock

Target Speed Quirk

- Some devices fail to bring up the Link at the maximum supported speed
  - LBMS=1 + DLLLA=0 used to identify these devices
  - Quirk brings the Link up at the lowest speed (gen1 2.5GT/s)
  - A device might be able switch back to a higher speed afterwards
- Changes Link Speed so should use the new API
- Integration challenges:
  - Currently monopolizes LBMS bit
  - Quirk runs very early, before pci_bus is exists and portdrv / services drivers
  - Can run later, bwctrl already setup
  - Usually on (disabling quirks behind EXPERT=y)

Target Speed Quirk and Bandwidth Controller Integration

Current approach:

PCI_ QUIRKS	PCIE_ BWCTRL	Phase	LBMS	bwctrl locking[1]
n[2]	n	–	–	none[3]
y	n	–	directly, no counting	none[3]
y	y	initial scan	directly, no counting	not probed yet, only rwsem
y	y	after initial scan	counted by bwctrl	rwsem+ per port lock

[1] Variations hidden by locking wrappers

[2] Requires EXPERT=y

[3] RMW ops are still protected

Bandwidth Controller Kconfig Integration

- Internal API for changing Link Speed
  - Do users require it to be always available?
    - E.g., device known to fail if the Link Speed is higher than x.
  - Current approach: Only build bare minimum to support changing Link Speed
    - Does not track Link Speed changes other times
- Alternative way out: hide bwctrl disable behind EXPERT=y?