CXL 2.0 + Linux + QEMU = Yes

CXL 2.0 + Linux + QEMU = ~~Yes~~

Introduction

Last slide first!!!

Agenda

- Introduction
- CXL 2.0 Background
  - Linux Driver Details
- QEMU
- Future

Intro/Links

- Communications
  - #cxl on OFTC
  - linux-cxl@vger.kernel.org
- Drivers
  - https://git.kernel.org/pub/scm/linux/kernel/git/cxl/cxl.git/
- QEMU
  - https://gitlab.com/bwidawsk/qemu
  - https://github.com/pmem/run_qemu
- Userspace
  - https://github.com/pmem/ndctl/tree/cxl-2.0v3
  - https://gitlab.com/bwidawsk-cxl/cxl_rs

CXL 2.0 Background

CXL Features

Challenges

Industry trends driving demand for faster data processing and next-gen data center performance

Increasing demand for heterogeneous computing and server disaggregation

Need for increased memory capacity and bandwidth

Lack of open industry standard to address next-gen interconnect challenges

CXL

An open industry-supported cache-coherent interconnect for processors, memory expansion and accelerators

Coherent Interface

Leverages PCIe® with 3 mix-and-match protocols

Low Latency

.Cache and .Memory targeted at near CPU cache coherent latency

Asymmetric Complexity

Eases burdens of cache coherent interface designs

CXL Features

Challenges

Industry trends driving demand for faster data processing and next-gen data center performance

Increasing demand for heterogeneous computing and server disaggregation

Need for increased memory capacity and bandwidth

Lack of open industry standard to address next-gen interconnect challenges

CXL

An open industry-supported cache-coherent interconnect for processors, memory expansion and accelerators

Coherent Interface

Leverages PCIe® with 3 mix-and-match protocols

Low Latency

.Cache and .Memory targeted at near CPU cache coherent latency

Asymmetric Complexity

Eases burdens of cache coherent interface designs

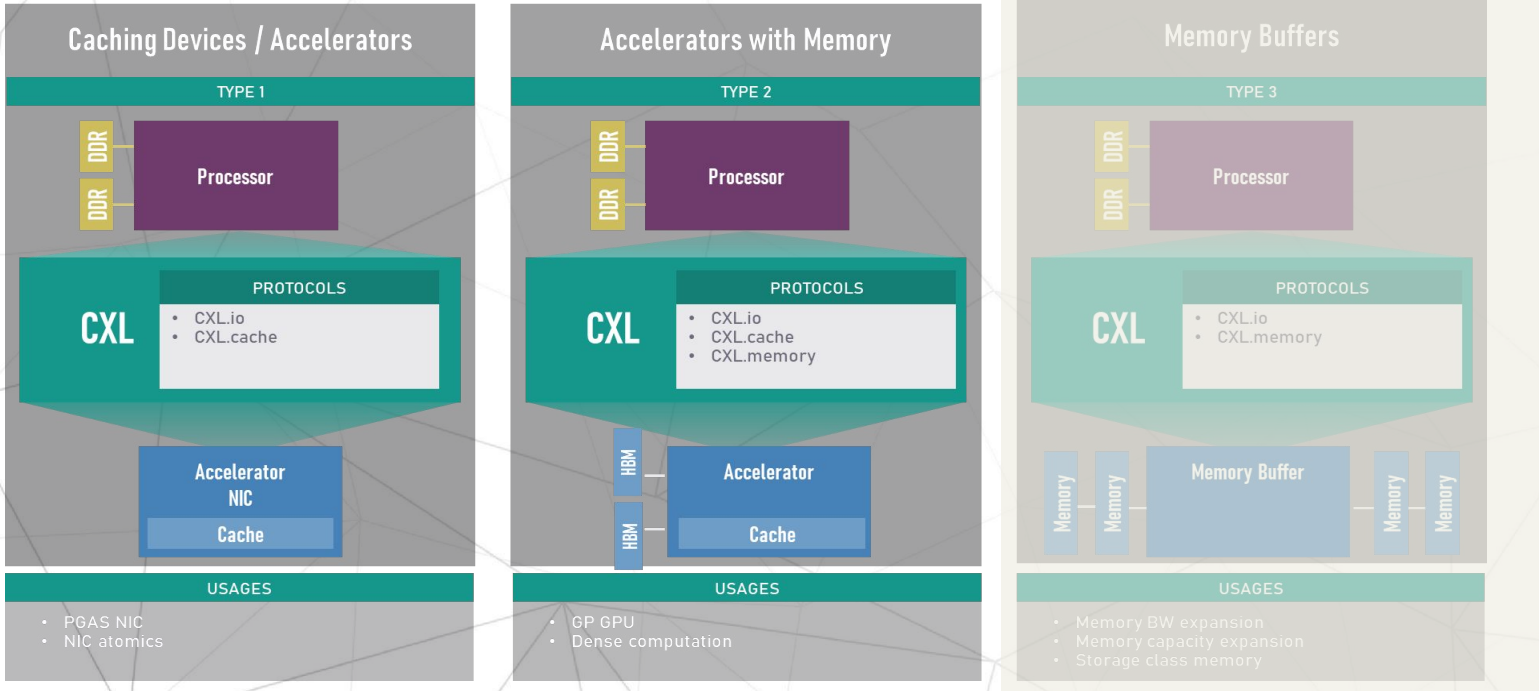
Who Cares?

Alibaba Group
AMD
arm
cisco
DELL EMC
facebook
Hewlett Packard Enterprise
HUAWEI
Google
IBM
intel
MICROCHIP
Microsoft
NVIDIA
XILINX

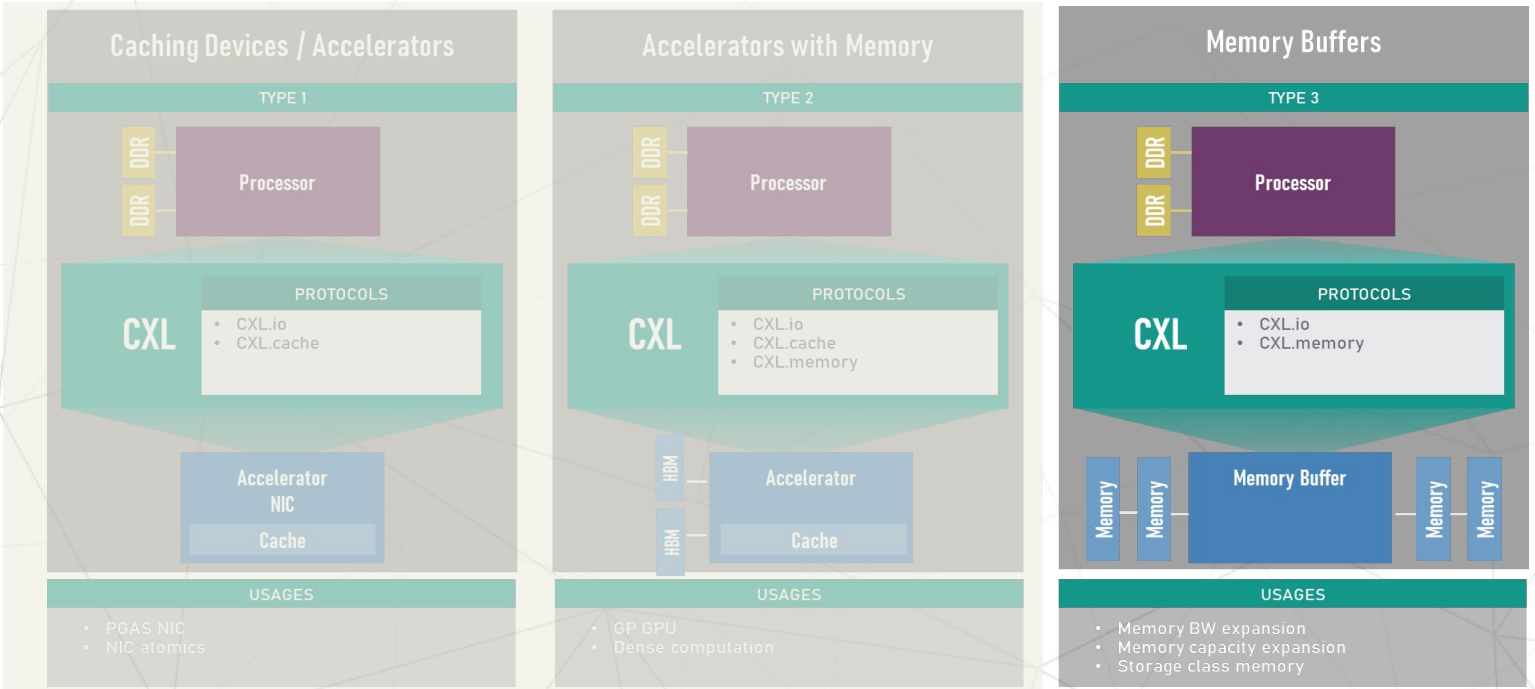
CXL Board of Directors

CXL Compute Express Link™

Usage Models

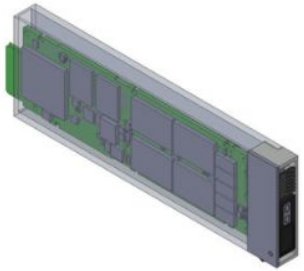


Usage Models

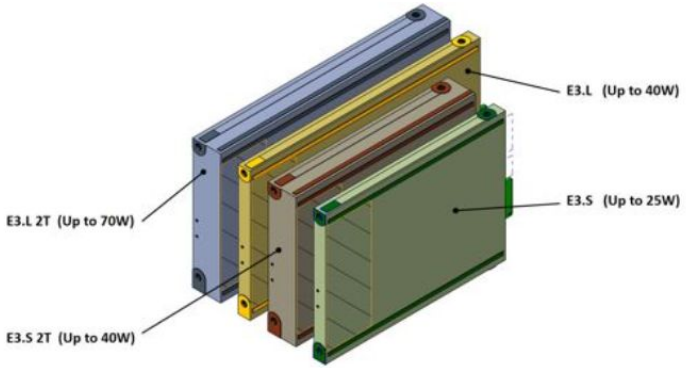


Form Factors

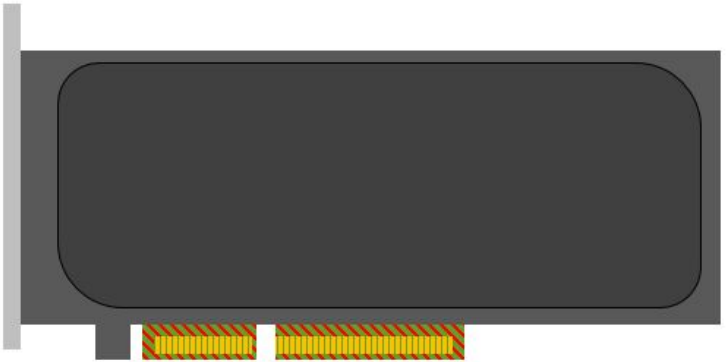
Ex #1: EDSFF E1.S



Ex #2: EDSFF E3.S / E3.L



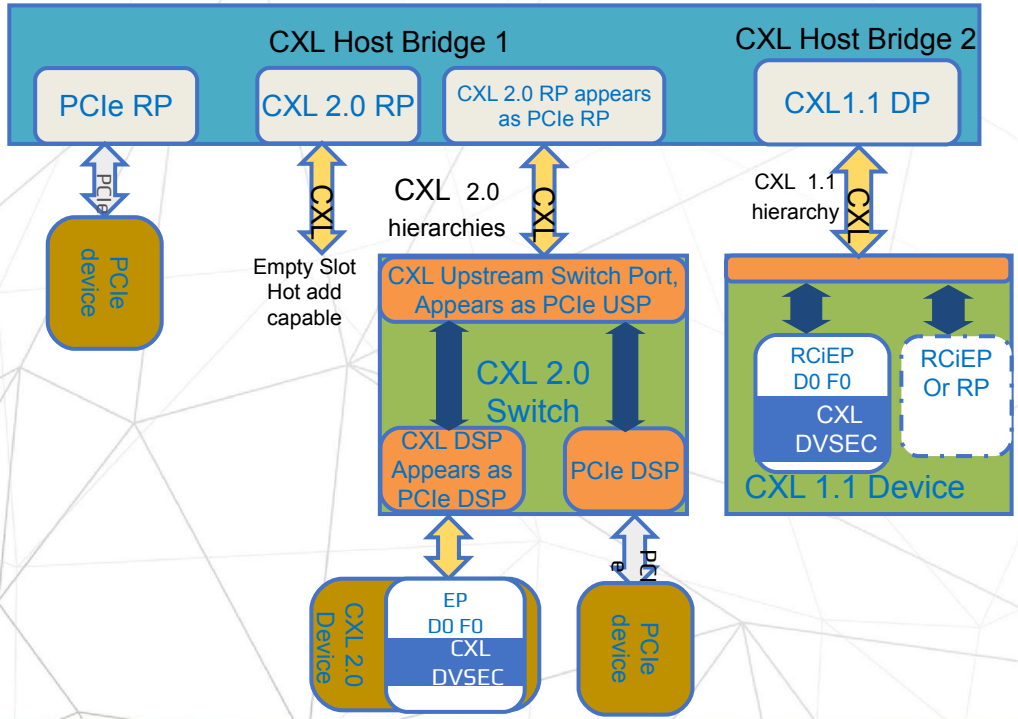
Ex #3: Add-in Card (AIC)



Factors	EDSFF E1.S	EDSFF E3.S / E3.L	AIC
Area vs. DDRx Server DIMM	• Smaller	• Larger	• Larger (larger than E3.S/L)
Expected Max. Power Range	• 12 ~ 25W	• 25W ~ 40W (1T), 40W ~ 70W (2T)	• Similar range compared to E3.S/L

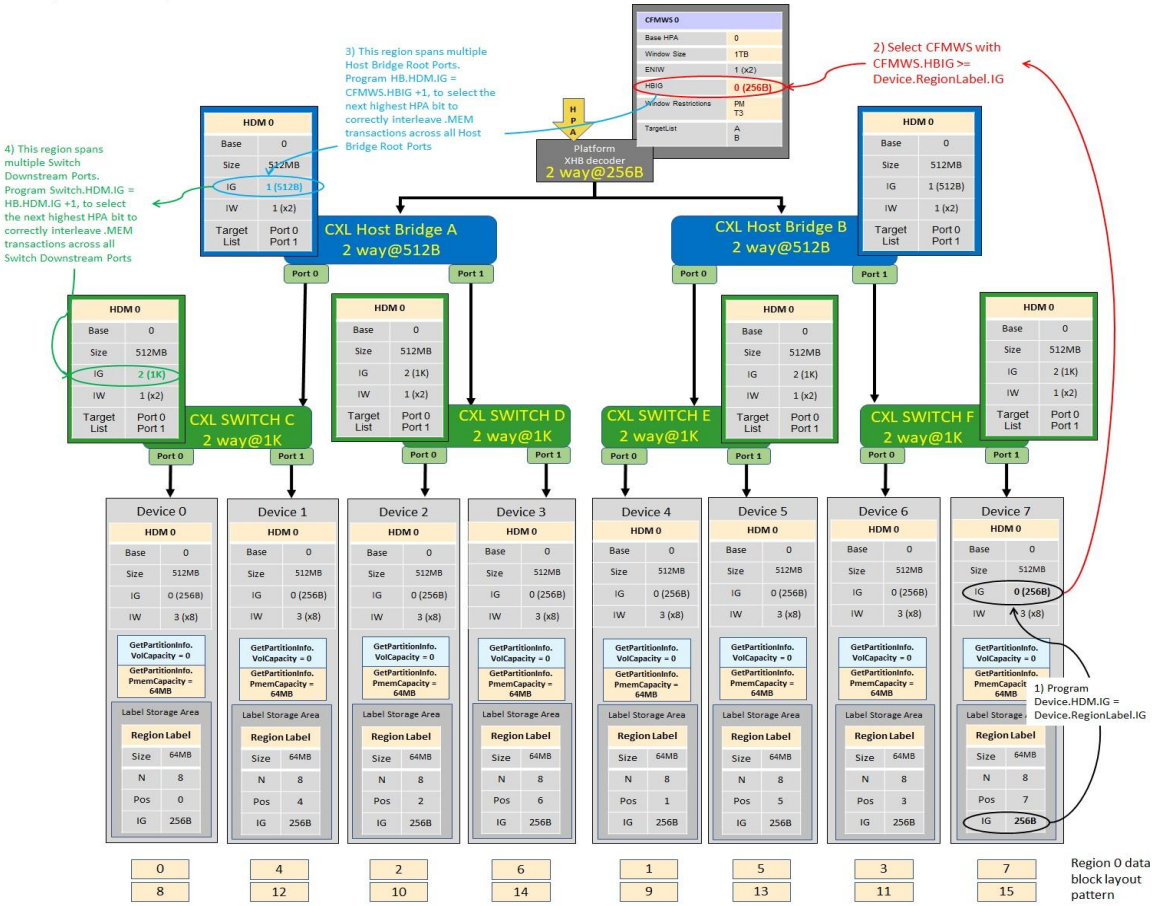
Reference: snia.org

CXL Topology



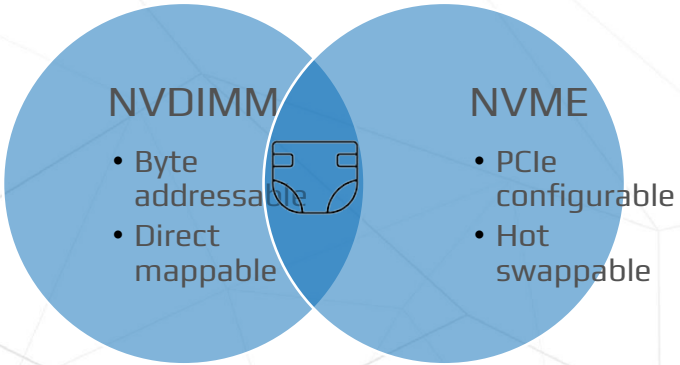
- CXL 2.0 hierarchy appears like PCIe hierarchy
  - Legacy PCI SW and CXL SW sees a RP or DSP with Endpoints below
  - CXL link/interface errors are signaled to RP, not RCEC
  - Port Control Override registers prevent legacy PCIe software from unintentionally resetting the device and the link
- Interleaving
  - Cross host bridge
  - Switch
  - Device

Interleave


CFMWS 0
Base HPA 0
Window Size 1TB
ENIW 1 (x2)
HBIG 0 (256B)
Window Restrictions PM T3
TargetList A B
2) Select CFMWS with CFMWS.HBIG >= Device.RegionLabel.IG
3) This region spans multiple Host Bridge Root Ports. Program HB.HDM.IG = CFMWS.HBIG +1, to select the next highest HPA bit to correctly interleave .MEM transactions across all Host Bridge Root Ports
4) This region spans multiple Switch Downstream Ports. Program Switch.HDM.IG = HB.HDM.IG +1, to select the next highest HPA bit to correctly interleave .MEM transactions across all Switch Downstream Ports
HPA
Platform XHB decoder 2 way@256B
HDM 0
Base 0
Size 512MB
IG 1 (512B)
IW 1 (x2)
Target List Port 0 Port 1
CXL Host Bridge A 2 way@512B
CXL Host Bridge B 2 way@512B
Port 0
Port 1
IG 2 (1K)
CXL SWITCH C 2 way@1K
CXL SWITCH D 2 way@1K
CXL SWITCH E 2 way@1K
CXL SWITCH F 2 way@1K
Device 0
Device 1
Device 2
Device 3
Device 4
Device 5
Device 6
Device 7
IG 0 (256B)
IW 3 (x8)
GetPartitionInfo. VolCapacity = 0
GetPartitionInfo. PmemCapacity = 64MB
Label Storage Area
Region Label
Size 64MB
N 8
Pos 0
Pos 4
Pos 2
Pos 6
Pos 1
Pos 5
Pos 3
Pos 7
IG 256B
1) Program Device.HDM.IG = Device.RegionLabel.IG
0 8
4 12
2 10
6 14
1 9
5 13
3 11
7 15
Region 0 data block layout pattern

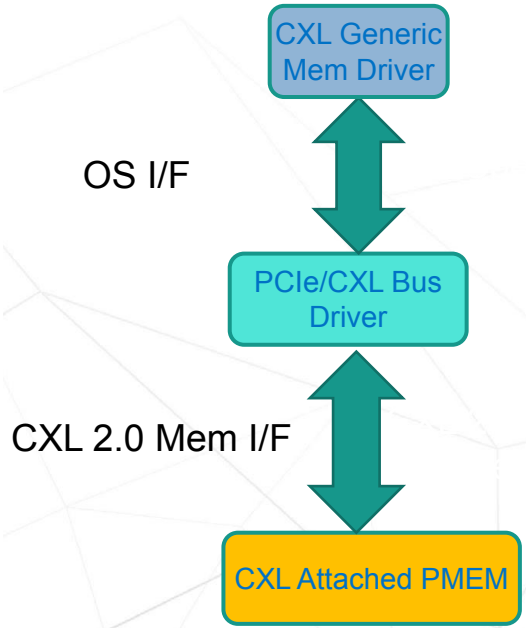
CXL Persistent Memory Devices

PCI and NVDIMM had a coherent byte addressable baby… with atomics.



PMEM

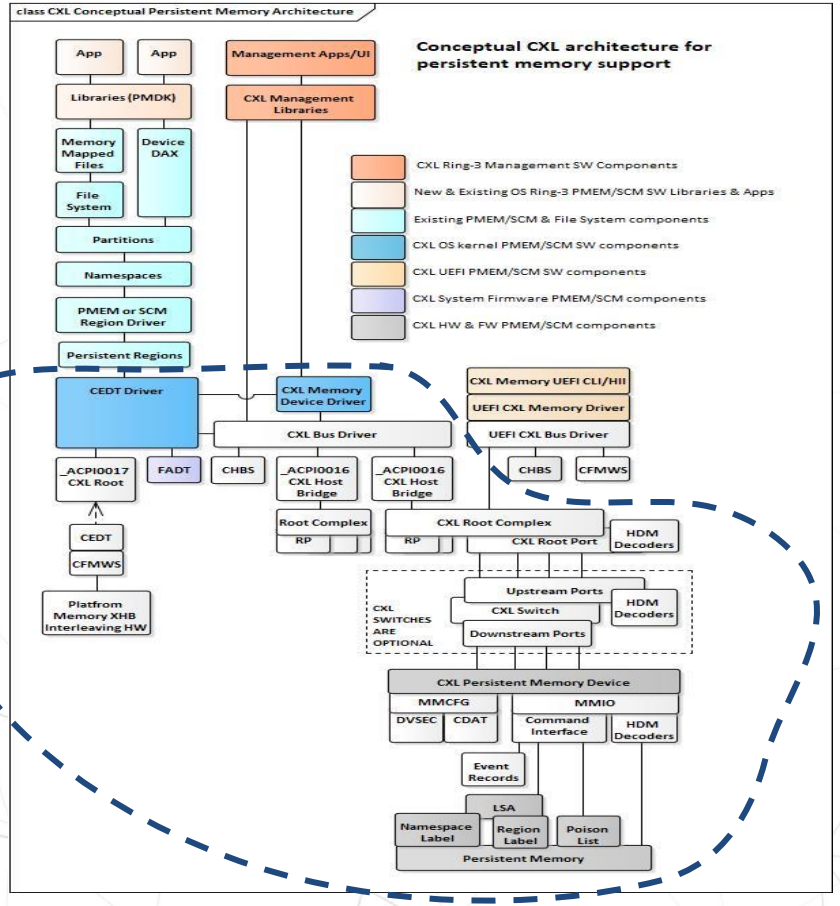
- Persistent memory devices rely on System Software for provisioning and management
- CXL 2.0 introduces a standard register interface
- A generic memory device driver simplifies software enabling
- Architecture Elements
  - Defined as number of discoverable Capabilities
  - Capabilities includes Device Status and standard mailboxes, accessed via MMIO registers
  - Standardized mailbox commands that cover errors/health, alerts, partitioning, passphrases etc.
  - Allow Vendor specific extensions



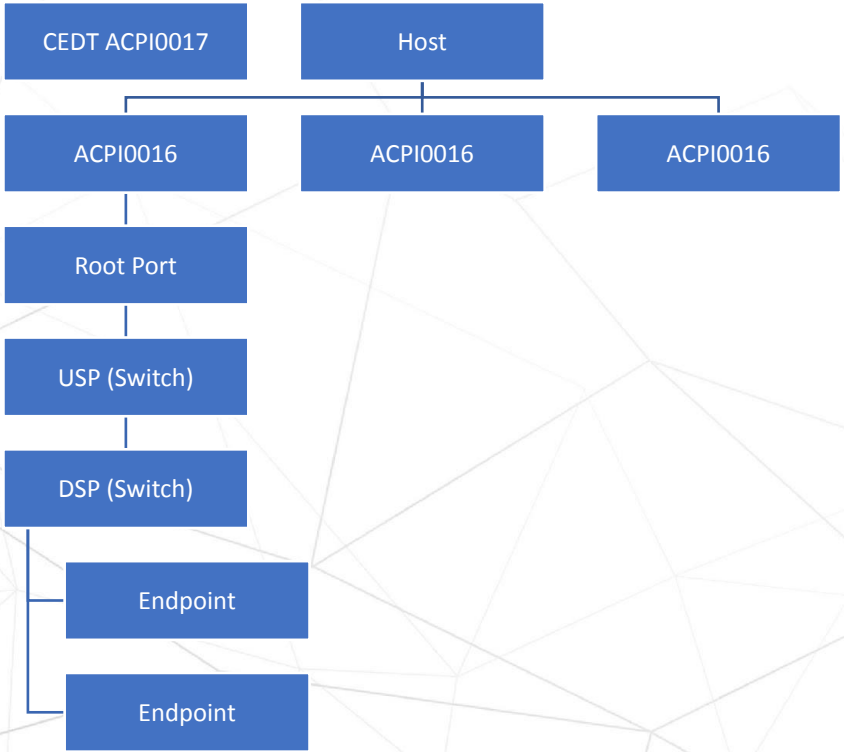
LINUX PLUMBERS CONFERENCE September 20-24, 2021

Linux Driver Details

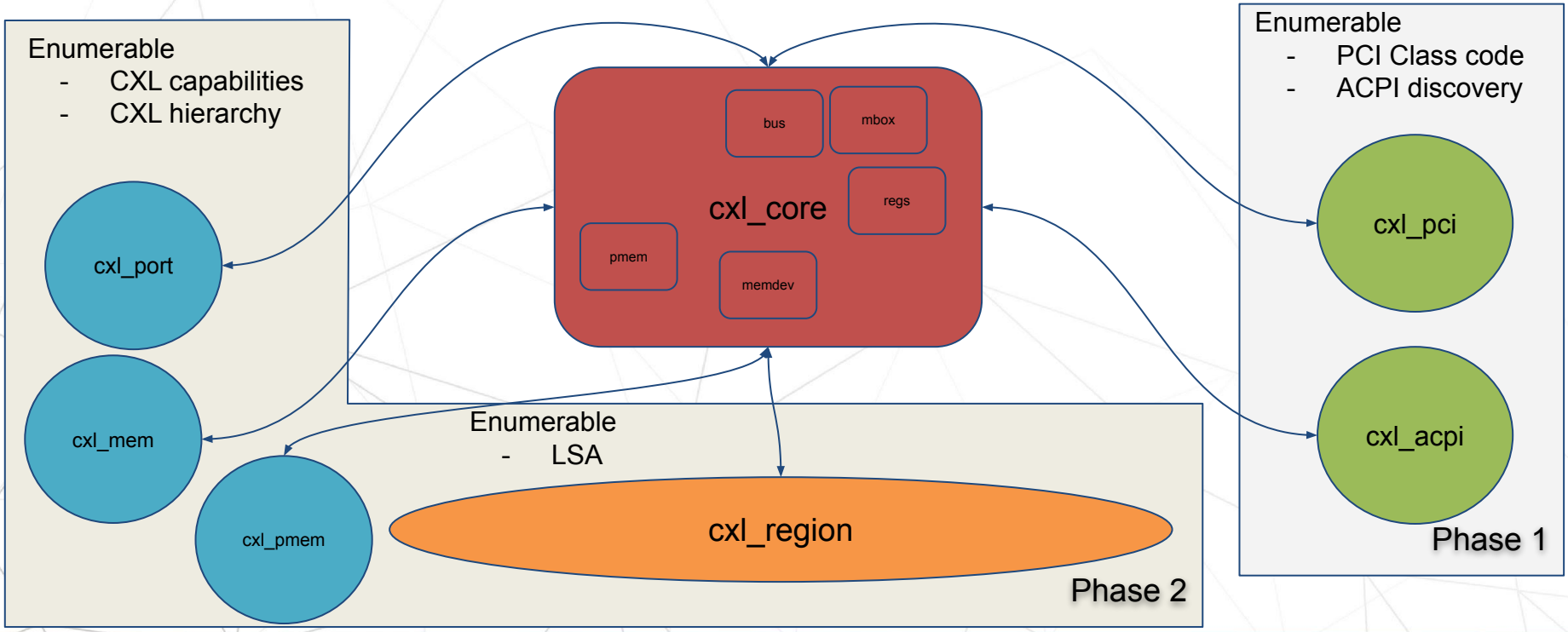
Software Responsibilities



SW Enumerable Components



Linux Drivers



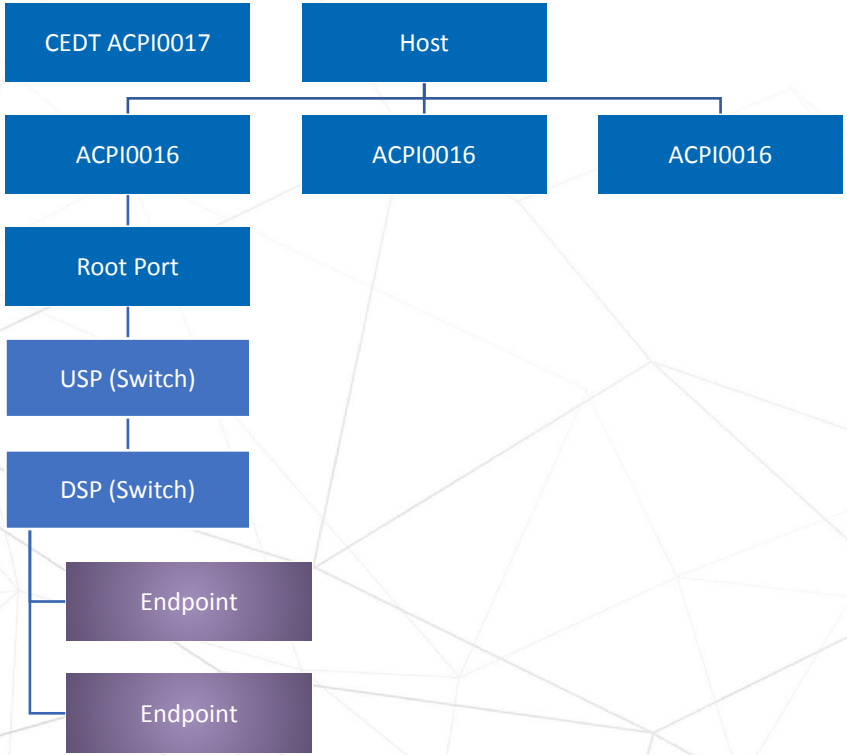
cxl_core

- Maintains cxl_driver infra
- Interfaces with LIBNVDIMM
- Manages device (sysfs)
  - Services to add devices, ie. cxl_decoder_add()
- IOCTL interface
- Common functionality
  - Mailbox controls (session layer)
  - Register mapping

cxl_pci

- Probed like a typical PCI device
  - `{ PCI_DEVICE_CLASS((PCI_CLASS_MEMORY_CXL << 8 | CXL_MEMORY_PROGIF), ~0)}`
- CXL device manageability
  - Implements mailbox transport (CXL) protocol
- Enumerates CXL device for subsequent driver
  - cxl_mem can’t run until cxl_pci is done
- Attestation/Security/Whatever

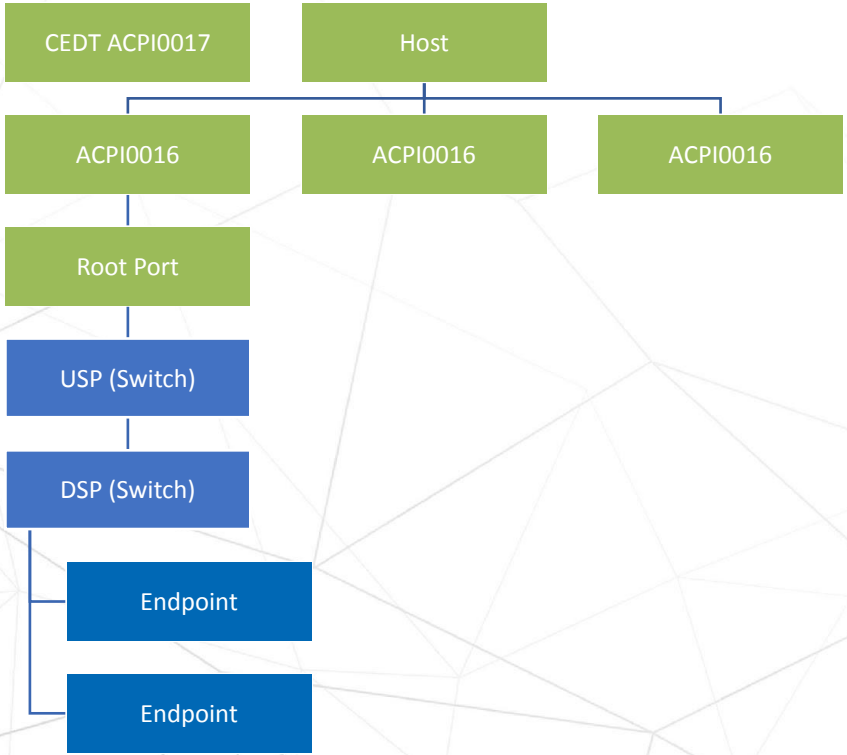
cxl_pci


CEDT ACPI0017
Host
ACPI0016
ACPI0016
ACPI0016
Root Port
USP (Switch)
DSP (Switch)
Endpoint
Endpoint

cxl_acpi

- Probed like a typical ACPI device
  - `{ "ACPI0017", (unsigned long) &native_acpi0017 },`
- Platform specific CXL enumeration
  - Mostly specified in UEFI, and CXL
- ACPI0017 starts enumeration of CXL ports
  - CEDT
  - “Root level” ports (platform)
  - Hostbridges and root ports

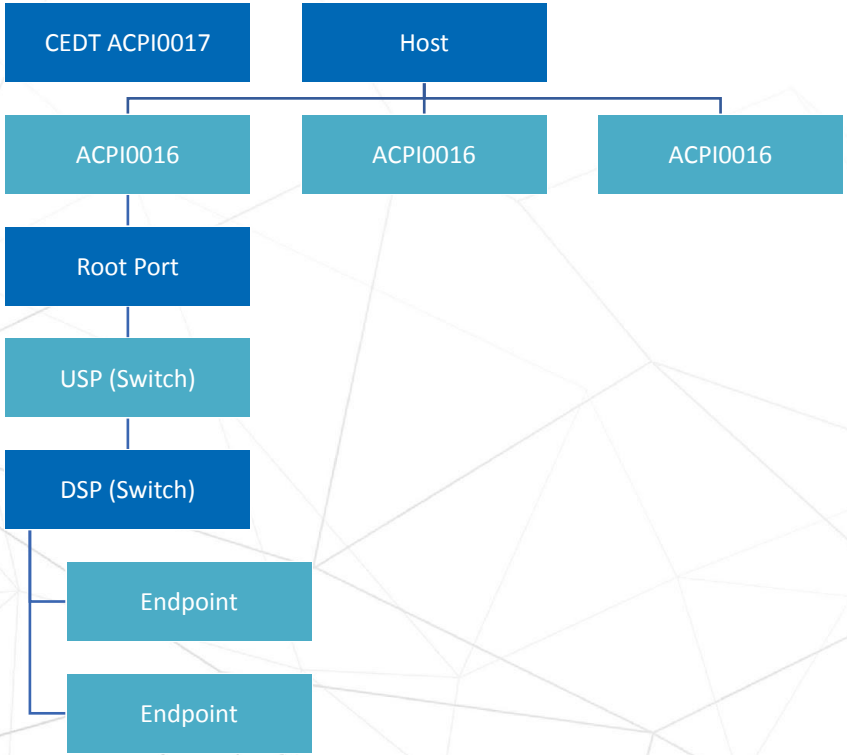
cxl_acpi


CEDT ACPI0017
Host
ACPI0016
ACPI0016
ACPI0016
Root Port
USP (Switch)
DSP (Switch)
Endpoint
Endpoint

cxl_port

- Ports are created for all components with an “upstream port”
  - Hostbridge
  - Switch
  - Endpoint
- Enumeration and control and control of decoder resources
  - Provides as a service for other drivers

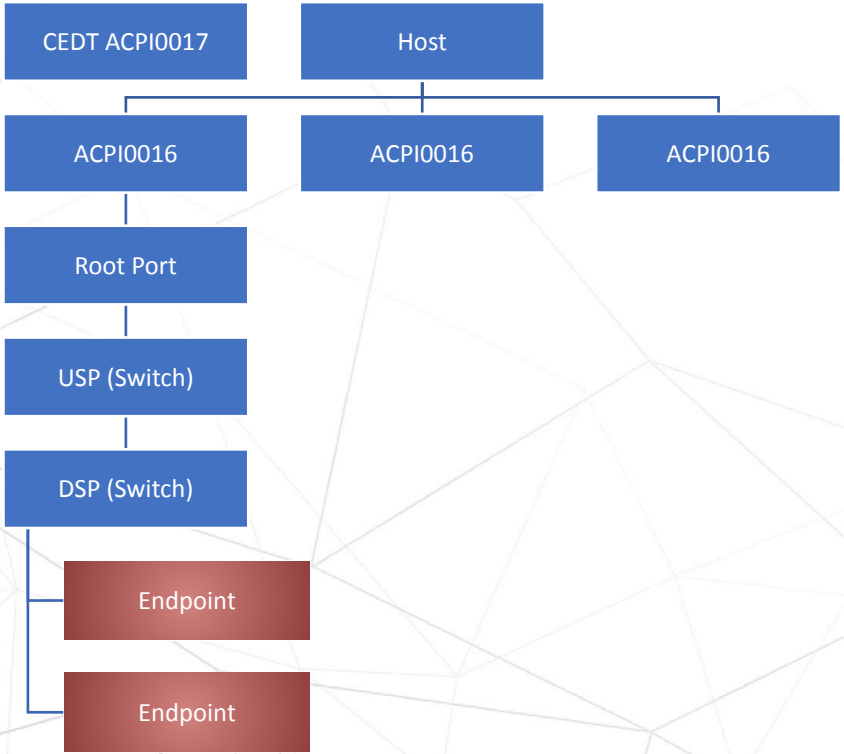
cxl_port


CEDT ACPI0017
Host
ACPI0016
ACPI0016
ACPI0016
Root Port
USP (Switch)
DSP (Switch)
Endpoint
Endpoint

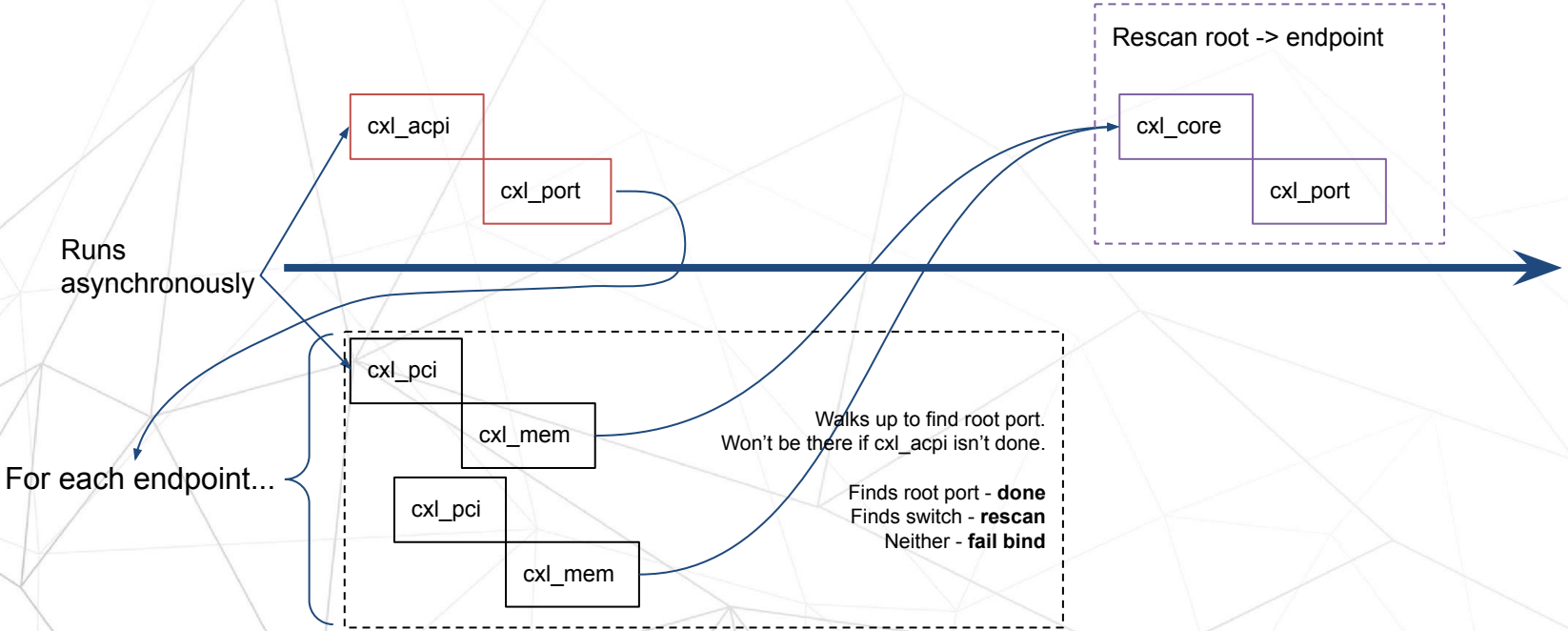
cxl_mem

- connects a device enumerated with cxl_pci to CXL.mem.
  - Implements device functionality not handled by cxl_pci
- “exports” if device is CXL.mem routed and enabled

cxl_mem


CEDT ACPI0017
Host
ACPI0016
ACPI0016
ACPI0016
Root Port
USP (Switch)
DSP (Switch)
Endpoint
Endpoint

Enumeration Timeline

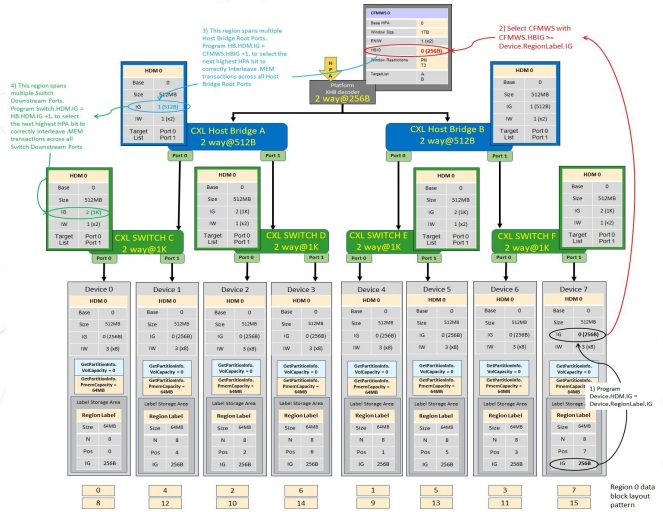


Volatile vs. Persistent

		Persistent	Volatile	
1	Configured by BIOS	✗	✓	• BIOS configures all volatile capacities • BIOS may check PMEM devices, but will not configure regions. ○ BIOS may configure bootable PMEM
2	OS managed	✓	☺	• OS initializes persistent regions • OS may create new persistent regions • Manages hotplug, error, and reset sequences for both
3	Requires CXL.mem capability	✓	✓	

Regions

- Region
  - Interleave set of devices
  - Parameters (IG, HPA, etc)
  - Stored in the Label Storage Area
- Creation
  - Via sysfs ABI
  - Provisioned offline
    - Manufacture time
- Responsibilities
  - Validating region configuration
  - Programming HDM decoders

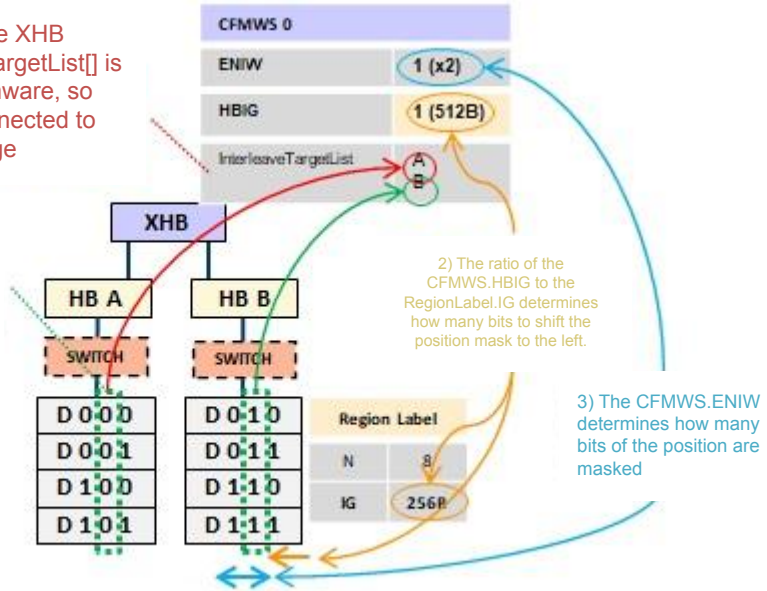


Region Validation

1) The ordering of the XHB CFMWS.InterleaveTargetList[] is fixed by System Firmware, so devices must be connected to the correct host bridge

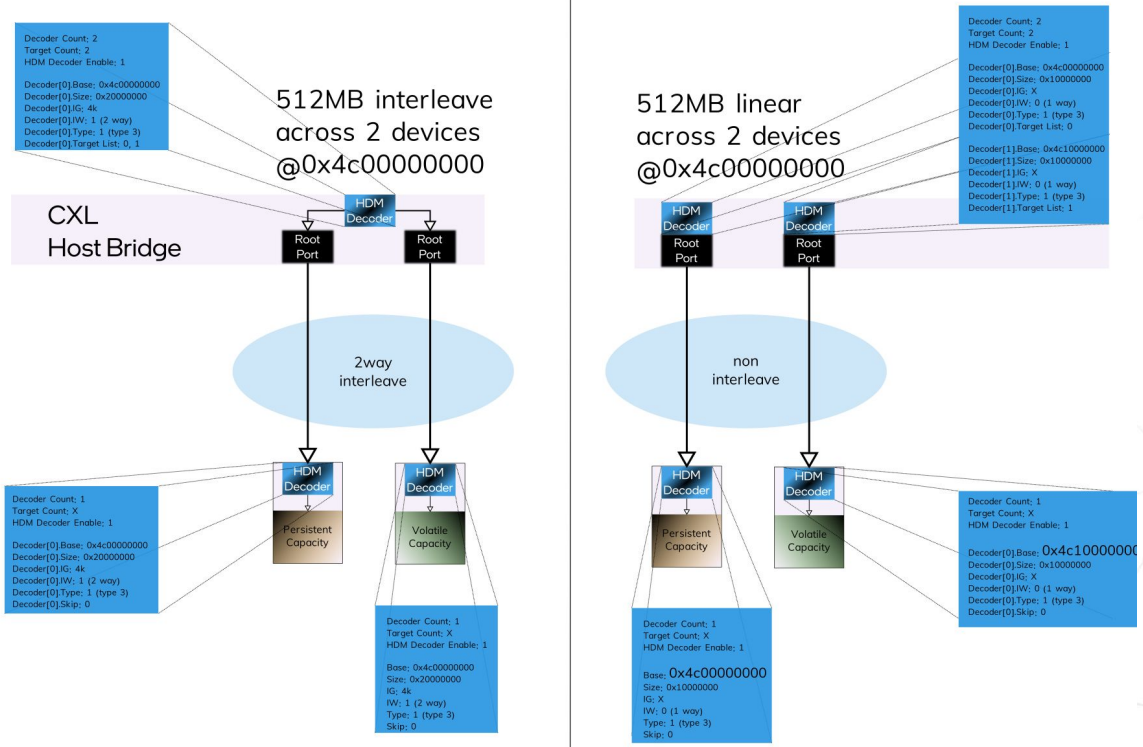
Verify each device has the same masked position value

```
Device[x].RegionLabel.Position >> (CFMWS.HBIG -
Device[x].RegionLabel.IG) & ((2^CFMWS.ENIW) - 1)
```



Valid XHB device configuration

Decoder Programming



Linux Interfaces

- Sysfs
  - /sys/bus/cxl/
- IOCTL
  - QUERY
  - SEND
  - Managed command set
  - RAW escape command
- Future
  - Region Creation ABI

LINUX PLUMBERS CONFERENCE September 20-24, 2021

QEMU

Review Goals

1) Upstream Linux Driver
   a) 0 days of spec release (v5.12)
      i) Ease customer adoption
   b) Backportable
   c) Platform aiding hw implementation and validation
      i) Validate the spec for driver usage
2) Reusable past driver bringup
   a) infra for regression testing
   b) Virtualization
3) Scalable
   a) Community Contributions & Fixes

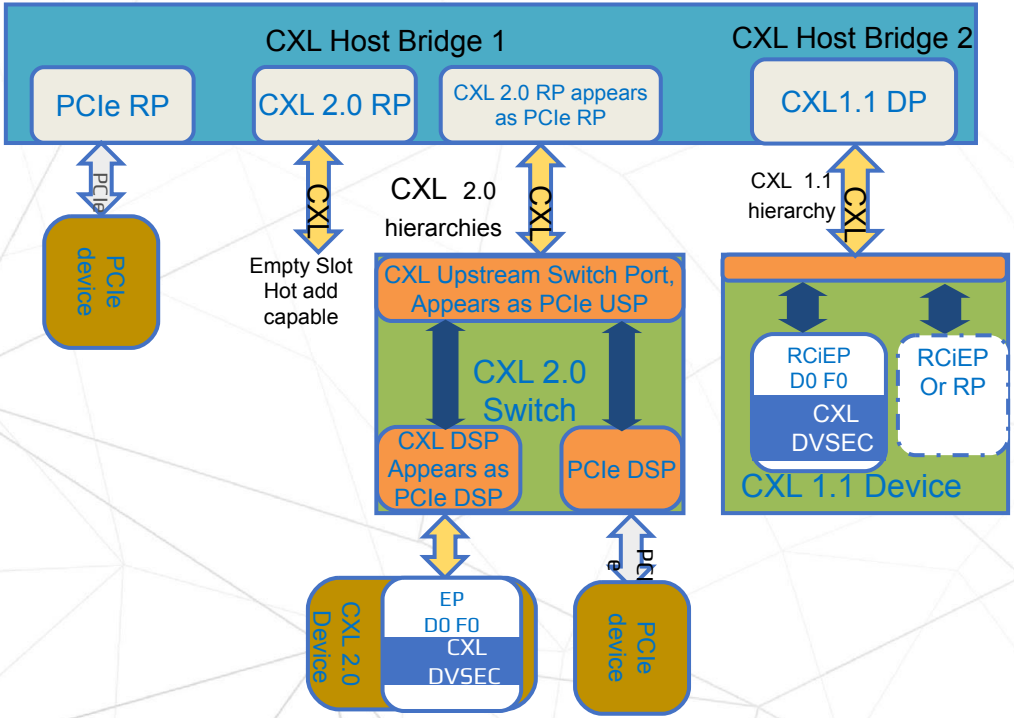
Pre-silicon State of the Art

- Hardware
  - No 2.0 FPGAs available
  - 1.1 is limited use and not readily available.
- Internal Simulation
  - Delays/
  - Can’t work with community
- Prior art
  - nfit_test
  - QEMU CCIX patches

Pre-silicon State of the Art

- Hardware
  - No 2.0 FPGAs available
  - 1.1 is limited use and not readily available.
- Internal Simulation
  - Delays/
  - Can’t work with community
- Prior art
  - nfit_test
  - QEMU CCIX patches

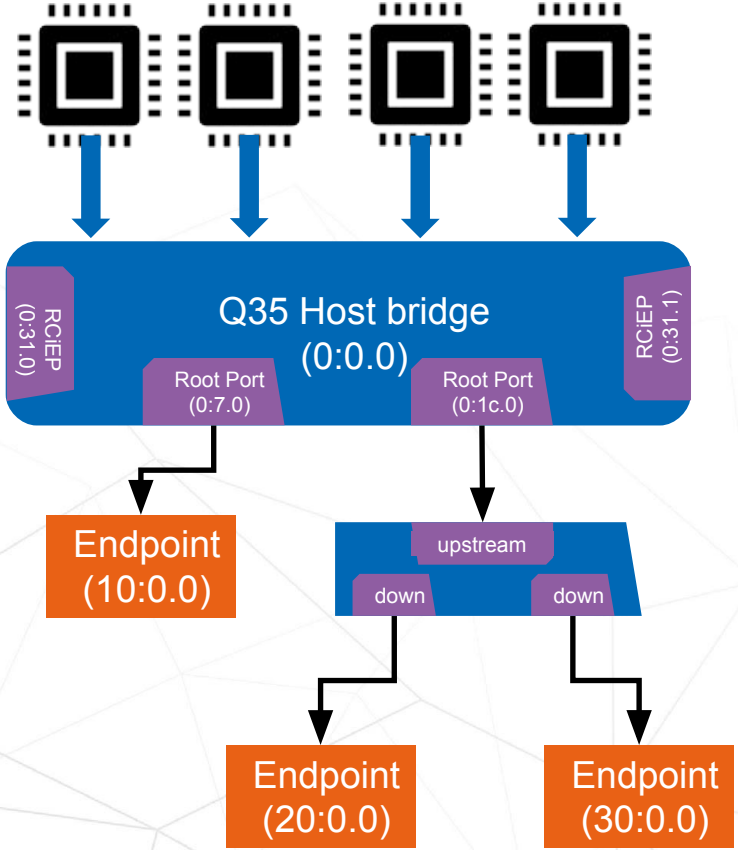
CXL Arch Review



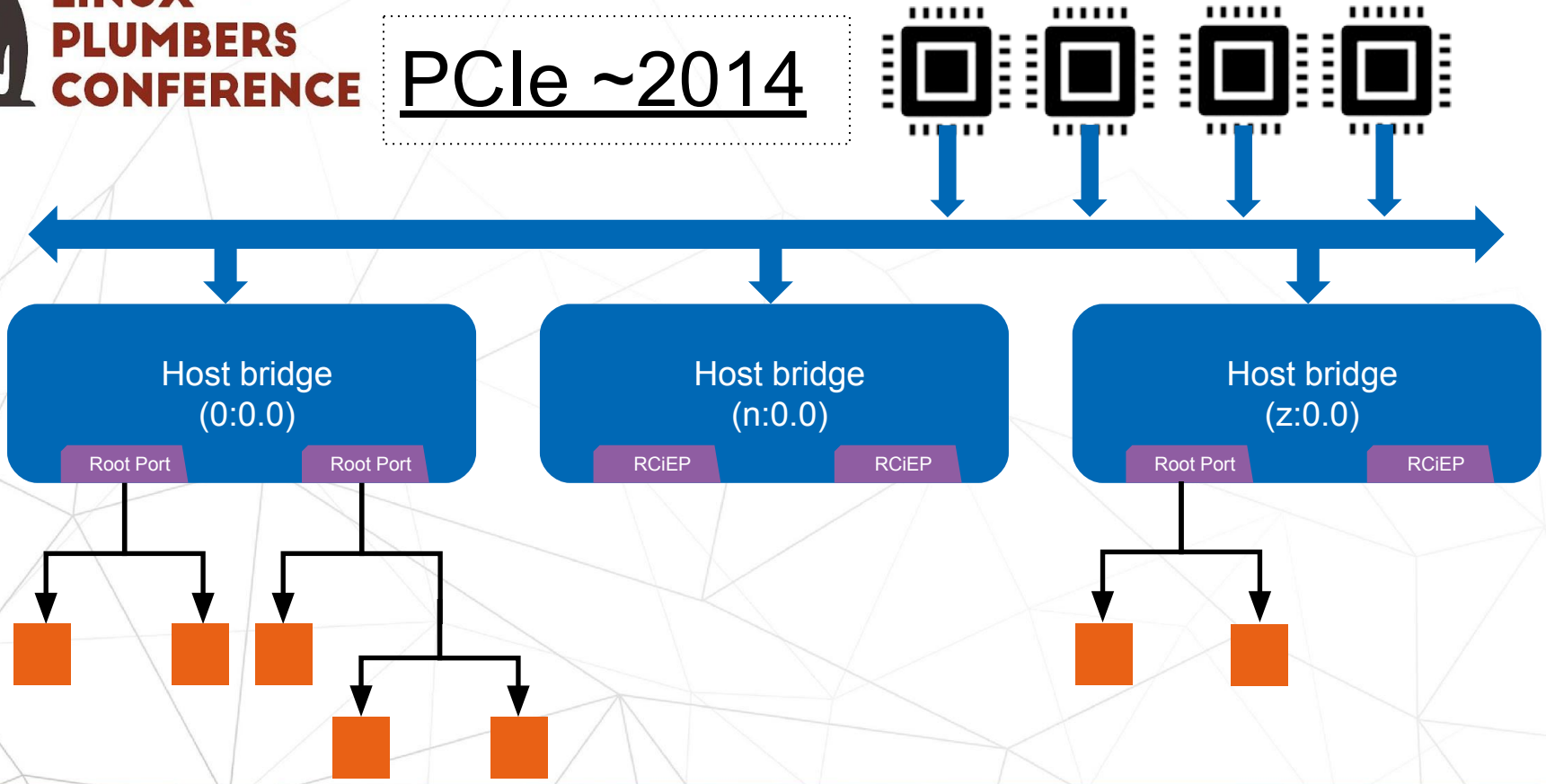
PCIe in QEMU

What we all know and love

- Single root complex
  - Endpoints
  - Root ports
  - Switches
- All traffic is funneled to the single host bridge
  - QPI/UPI (not modeled)



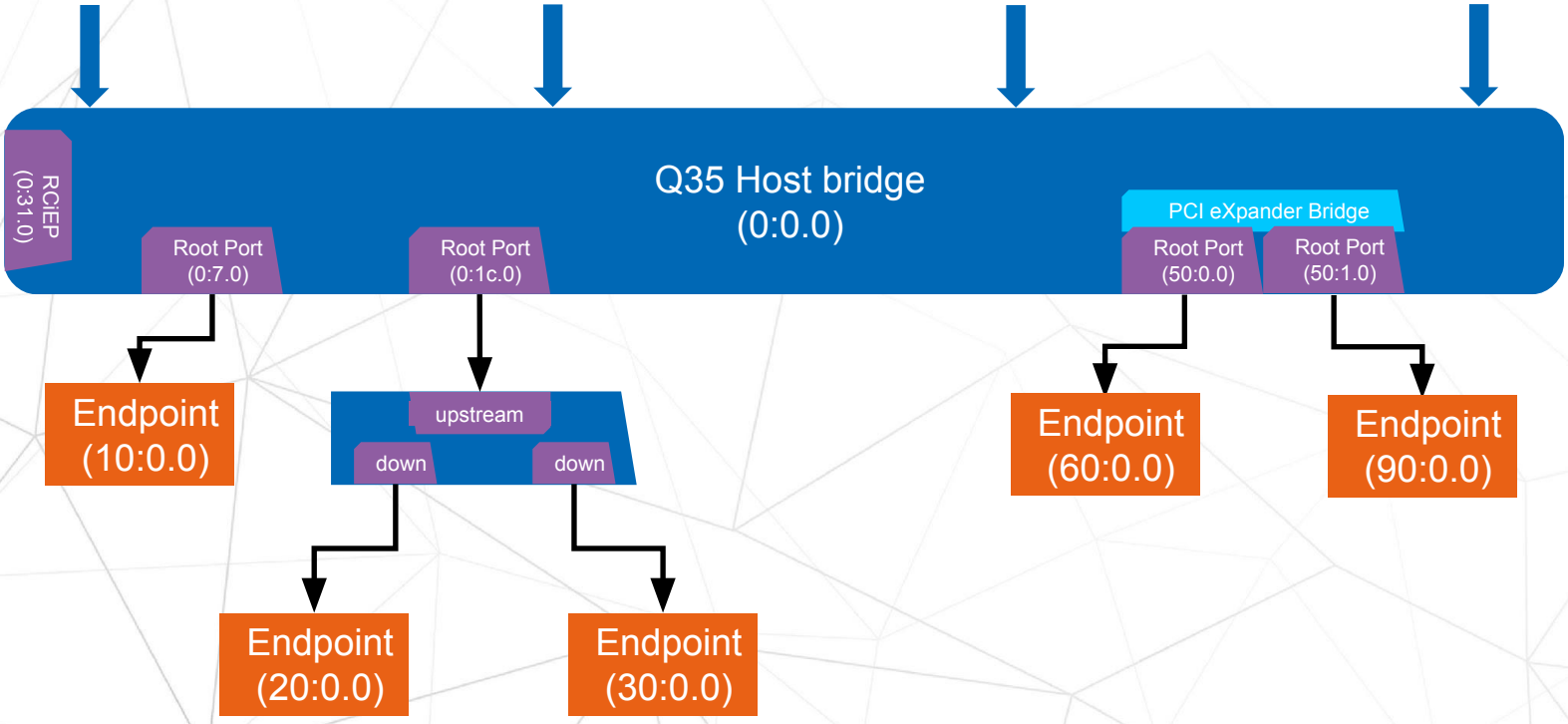
PCIe ~2014


Host bridge
(0:0.0)
Root Port
Root Port
Host bridge
(n:0.0)
RCiEP
RCiEP
Host bridge
(z:0.0)
Root Port
RCiEP

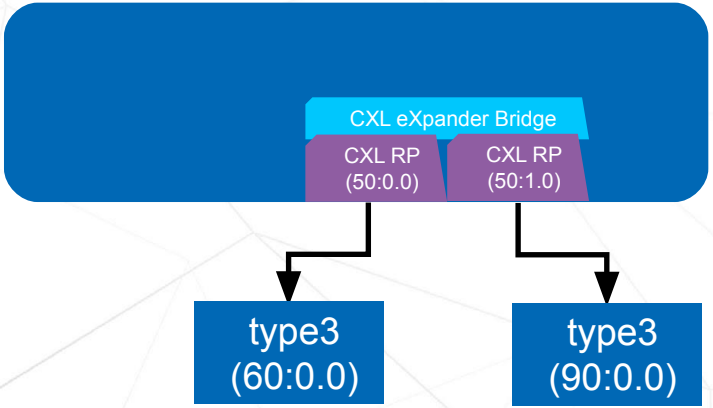
Options

- Hacks to make Q35 - CXL 2.0
  - Limited potential for interleave scenarios
  - Touching Q35 is risky.
  - Mistakes make everything work incorrectly.
- Replace Q35 with something newer
  - Still Risky.
  - A lot of work for not much gain
  - What good does modeling UPI do for QEMU?
  - Doubtful community wants it (support burden).

PCI eXpander Bridge (PXB)



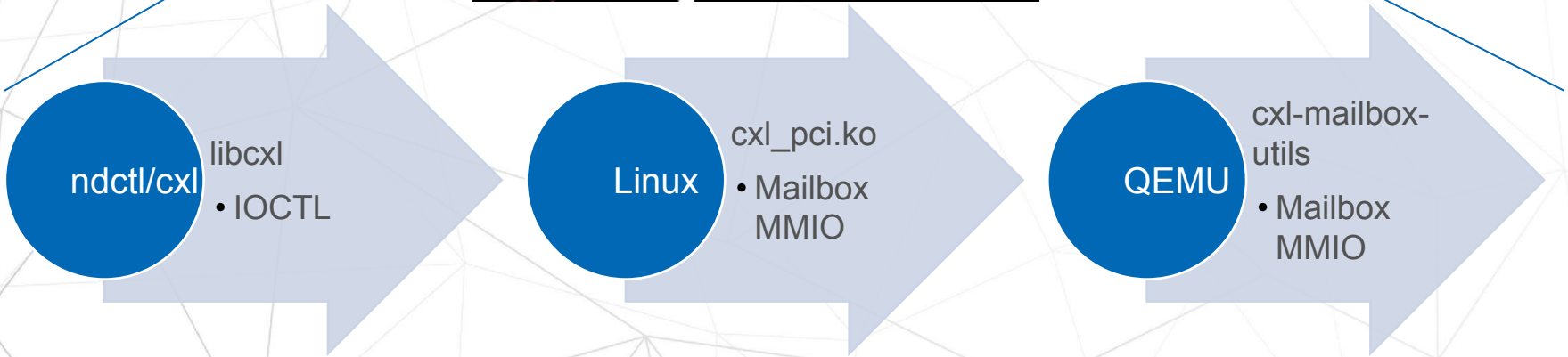
- CXL Type 3 device
  - hw/mem/cxl_type3.c
- CXL Root Port
  - hw/pci-bridge/cxl_root_port.c
- CXL PXB
  - hw/pci-bridge/pci_expander_bridge.c



- NVDIMM & PCI had a baby…
- Inherits from both interfaces
- Mailbox handling

```
static const TypeInfo ct3d_info = {
    .name = TYPE_CXL_TYPE3_DEV,
    .parent = TYPE_PCI_DEVICE,
    .class_init = ct3_class_init,
    .instance_size = sizeof(CXLType3Dev),
    .instance_init = ct3_instance_init,
    .instance_finalize = ct3_finalize,
    .interfaces = (InterfaceInfo[]) {
        { TYPE_MEMORY_DEVICE },
        { INTERFACE_CXL_DEVICE },
        { INTERFACE_PCIE_DEVICE },
        {}
    },
};
```

```
archlinux ~ # cxl list
[
  {
    "memdev":"mem0",
    "pmem_size":268435456,
    "ram_size":0,
    "fw_revision":"BWFW VERSION 00",
    "partition_align":0,
    "lsa_size":0
  }
]
```



November 10th 2020

Spec Released

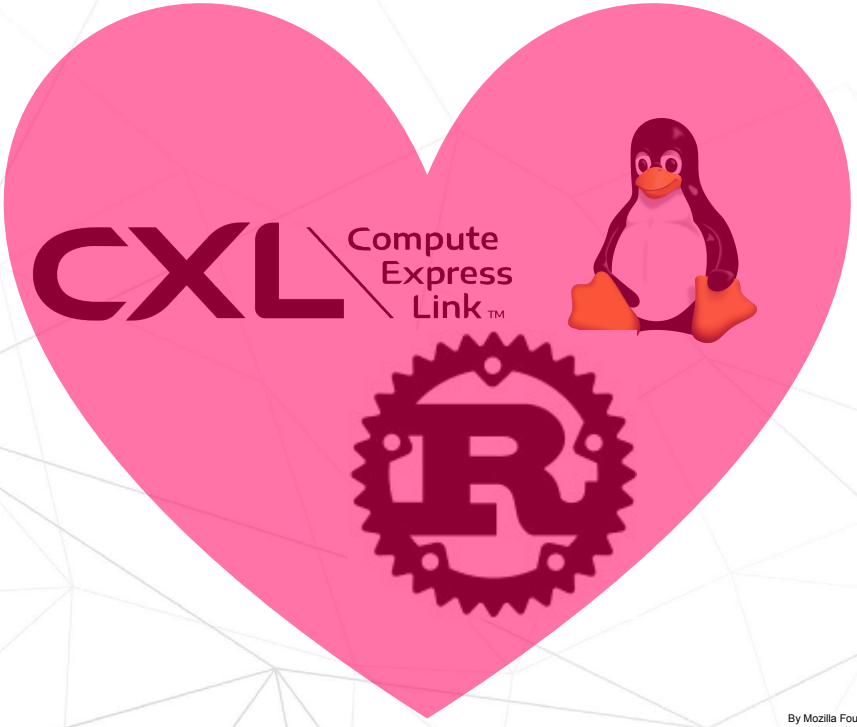
Linux Patches submitted

QEMU patches submitted

- QEMU v3 patches sent
  - v4 is ready for submission
  - Community contributions for DOE, CDAT, and SPDM
  - High bar for adding more
    - Nothing exists quite like a CXL memory device
      - Volatile + Persistent capacities
      - Interleaving at multiple levels
- Linux phase 1 driver merged in 5.12
  - Phase 2 actively being developed.
  - Community contributions for DOE and CDAT
- Spec issues found and the fixes prototyped in QEMU

Future

CXL RUST FTW!!!!!!!!!!



What to do

- Community didn’t adopt
  - Minimal feedback
  - Major reworks for interleave
  - cxl_test came along
- External contributions
    - DOE
    - CDAT
    - SPDM
- Commercial Adoption...


Let it go
Keep Alive
Ben
JAKE-CLARK.TUMBLR

We're Hiring...

https://jobs.intel.com/ShowJob/Id/3089754/Linux-Kernel-Development-Engineer

LINUX PLUMBERS CONFERENCE

September 20-24, 2021

- Linux
  - DPA mapping (WIP)
    - ~~Libnvdimm integration~~
  - Interleave
    - Provisioning
    - Recognition
  - Hotplug
    - Hot add
    - Managed remove
  - Asynchronous mailbox
- Userspace
  - Testing

- QEMU
  - Better tests
  - Upstream/Downstream Ports
  - Interleave Support
    - Host bridge
    - switch
  - More firmware commands
  - Hotplug support
  - Error testing
  - Interrupt support
  - Memory class device overhaul
  - ----------
  - Make Q35 CXL capable
  - CXL type 1 and 2 devices
  - CXL 1.1