

Compute Express Link(CXL), the next generation interconnect

-- Overview and the status of Linux --
Open Source Summit Japan 2023 ver.

2023/Dec/5

Yasunori Goto

Senior Software Engineer

Fujitsu Limited

E-mail:<y-goto at fujitsu.com>, X(Twitter): @YasunoriGoto1



- Yasunori Goto:

I have worked for Linux and related OSS since 2002

- Development for Memory hotplug feature of Linux Kernel
- Technical Support for troubles of Linux Kernel
- etc.

- Currently, a leader of Fujitsu Linux Kernel Development team

- For some years, I have mainly worked for Persistent Memory
 - [“The ideal and reality of NVDIMM RAS”](#)
 - China Linux Kernel Developer Conference 2018
 - [“The forefront of the development for NVDIMM on Linux Kernel”](#)
 - Linux Plumbers Conference 2021
- My team has been working on CXL since April 2023



- Please refer the CXL specification for proper understanding
 - Anyone can download the CXL specification from [official site](#)
 - What you need is only registering your name and e-mail address to the site
 - Though I tried to make sure there is no mistakes in my presentation, there might be misunderstandings or inaccuracies yet
 - The CXL specification and related specifications are very huge, and I could read only some parts of them
 - CXL 3.1: 1166page, PCI Gen6: 1923page, ACPI ver6.5: 1508page
 - If you can find mistakes, please let me know
- I recommend for you to study PCI/PCIexpress and ACPI specification beforehand
 - The CXL specification is too difficult if you don't know them

Anyway, I hope my presentation helps you to understand CXL!

- Overview of CXL specification until 2.0
- Additional specification of CXL 3.0 and 3.1
- The status of current Linux for CXL
 - Memory tiering
 - Memory hotplug/memory pool

Overview of CXL

Until 2.0

What is Compute Express Link

- A new specification of interconnect which connects devices like PCIe
 - CXL is abbreviation of **C**ompute **E**xpress **L**ink
 - [Official whitepaper says](#) “an open industry standard interconnect offering high-bandwidth, low-latency connectivity”
 - It is suitable to connect Smart devices like GPGPU, SmartNIC, FPGA, Computational Storage, and so on
 - In addition, it is also useful to expand memory (volatile memory and persistent memory)
 - The newest revision of specification is **3.1** which was released **2023/Nov/14th**
- CXL seems to be winner against other competing specifications
 - The Board of Directors of CXL includes numerous vendors and service providers
 - Alibaba, AMD, Arm, CISCO, DELL, Google, HPE, Huawei, IBM, Intel, Meta(Facebook), Microsoft, NVIDIA, Rambus, SAMSUNG
 - Other competing specifications seem not to be promising
 - Open CAPI and Gen-Z were assimilated into CXL
 - CCIX is not active
 - There is no new information after 2019 in CCIX Press Release
 - Since the Promoter companies of CCIX are also members of CXL, they can select CXL instead of CCIX

Why CXL becomes necessary?

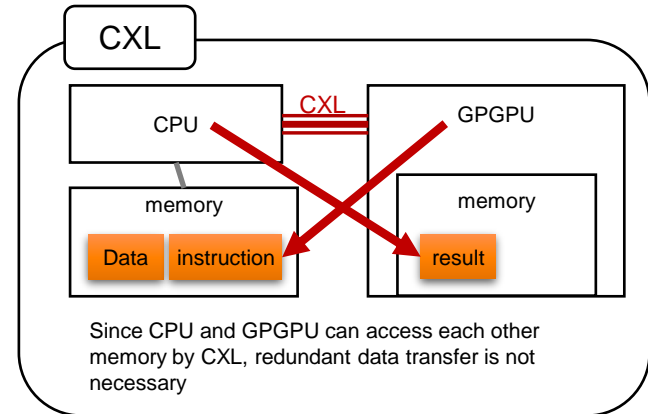
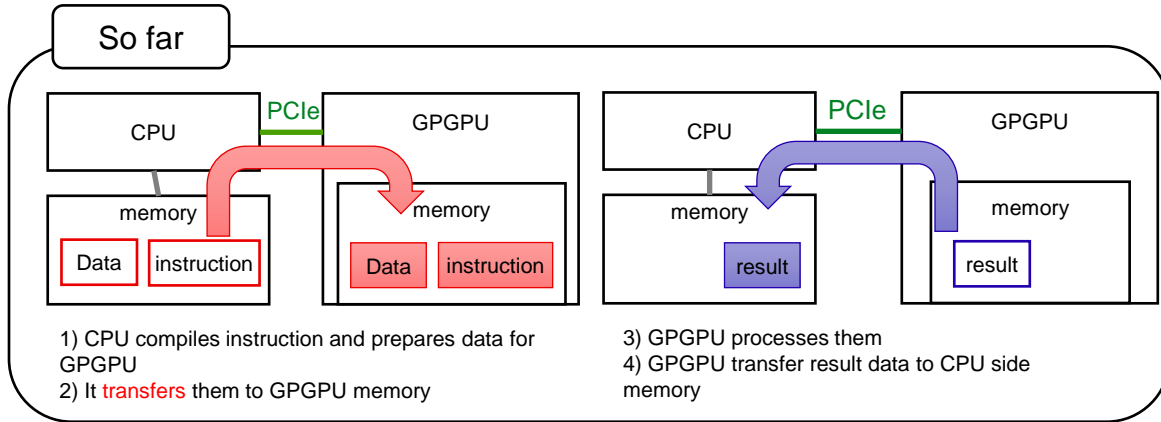
- Increasing demand for faster processing of data
 - Influence of current technology trends such as machine learning
- The need to offload processing due to reaching limitation of CPU performance enhancement
 - GPGPU、FPGA、Smart NIC must handle the processing instead
- Memory capacity must be increased
 - Though the number of CPU core increases, memory capacity does not follow it
 - Since DDR is a parallel interface, it is difficult to increase the number of CPU pins to connect more memory



New interconnect becomes necessary to connect devices and memories instead of PCI express or DDR

What is advantage of CXL?

- Example) Calculation of GPGPU will be more effective
 - So far, a CPU and a device must transfer data and instruction in bulk between DDR DRAM and GPGPU memory
 - Not only data, but also instructions for GPGPU must be transferred, it is a bit troublesome and needs time for the transfer
 - CXL allows that CPU and GPGPU can access other side's memory interactively
 - It will be effective for machine learning or any other modern analysis
 - Similar benefits can be obtained when you offload data processing to FPGA or SmartNICs



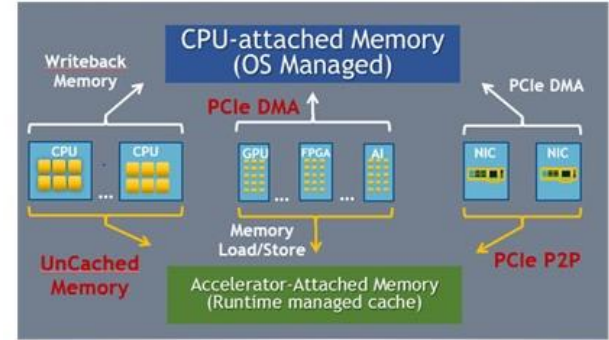
(*) CUDA Programming <https://www.sintef.no/globalassets/upload/ikt/9011/simoslo/evita/2008/seland.pdf>

To access each other's memory effectively

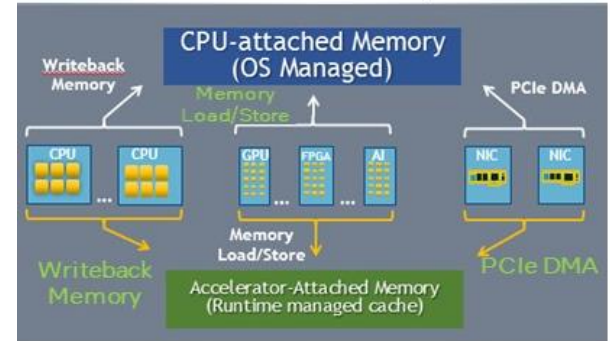
- Not only CPUs, but also devices require the use of cache to access memory from the other side **interactively**
 - Currently, PCIe does not allow the use of cache for transferring data
 - Even if the device memory is mapped to the host address space, CPUs must use "write-through" access, which is slow.
 - Devices need to transfer their data in bulk by DMA, which is not possible to **interactively** load and write memory
 - There are requirements against the above limitation
 - CPUs and Devices want to have **interactive** access using cache
 - They want to writeback their cache when it is necessary

CXL is created for the above requirements

Note: The right figure is quoted from the following (But the original figure seems to be keynote by Intel)
<https://www.servethehome.com/wp-content/uploads/2021/08/Intel-Hot-Interconnects-2021-CXL-1-Open-Interconnect.jpg>



With PCIe-only



CXL Enabled Environment

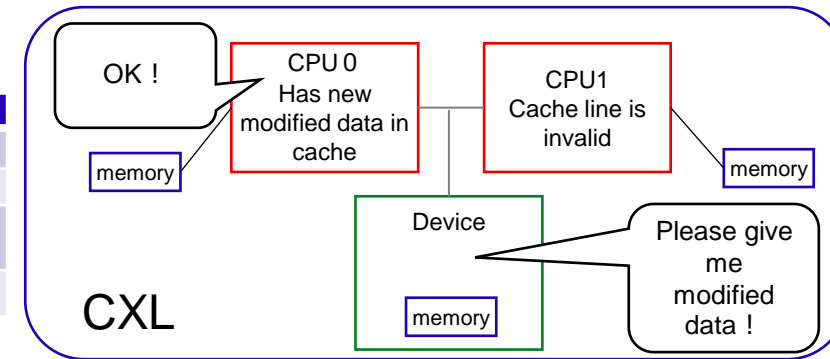
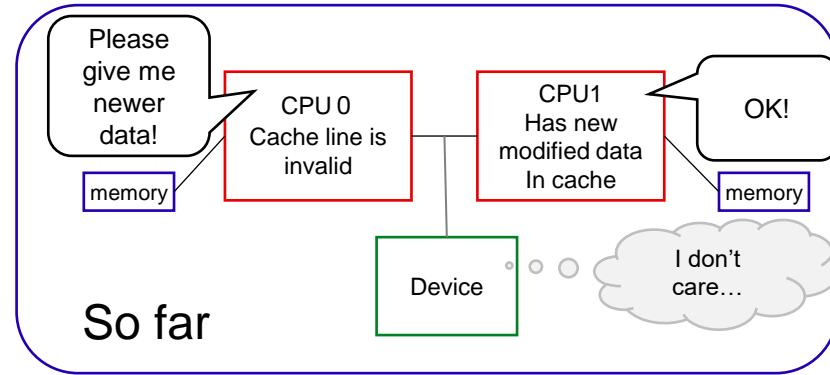
Cache-Coherent Interconnect

- [CXL Web page](#) says the following
 - “Compute Express Link™ (CXL™) is an industry-supported **Cache-Coherent** Interconnect for **Processors, Memory Expansion and Accelerators**. “

What is **Cache Coherent**, here?”

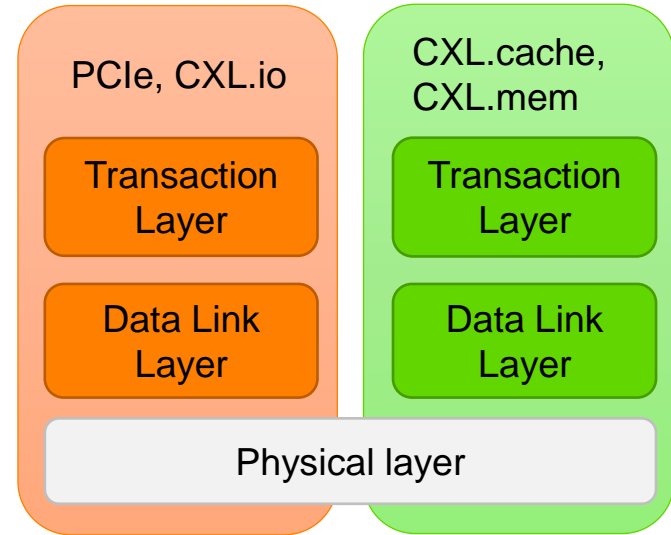
- So far, only CPUs must negotiate cache information for memory access each other
 - There are some famous protocols for cache coherency
 - E.g., MESI, MOESI
- To access each other between CPUs and devices with cache, the devices also need to coordinate cache information with CPUs
- CXL realizes it with MESI protocol

Status	Meaning
Modified	The cache line is present only in the current cache, and is dirty
Exclusive	The cache line is present only in the current cache, but is clean
Shared	This cache line is may be stored in other caches of the machine and is clean
Invalid	This cache line is invalid



Characteristic of CXL

- CXL utilizes PCIe spec. Gen 5.0 or later
 - Its Physical layer is same with PCIe
 - But, Upper layer becomes CXL original protocol
 - PCIe Gen 5.0 or later allows different protocol packet on its bus
- CXL protocol is mixture of the following 3 type protocols
 - CXL.io : used for CXL device detection, error report by PCIe way
 - CXL.cache : used to request, or communicate cache information between devices/accelerators(*) and CPUs
 - CXL.mem : used to request for memory access between devices/accelerators and CPUs
 - CXL.io is same protocol with PCIe, but others are new protocols of CXL



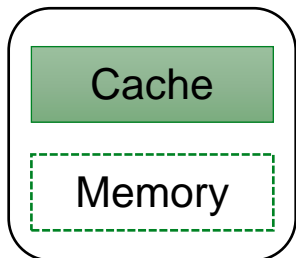
(*)Accelerator: Devices that may be used by software running on Host processors to offload or perform any type of compute, or I/O task

The 3 types of CXL device

- There are three definition of device type

Type-1

A device has cache and does not have memory, or its inside memory is not shown to host



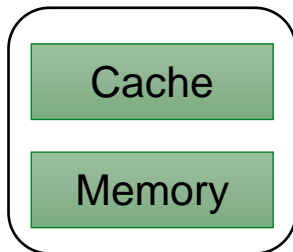
(e.g., SMART NIC or FPGA which has the above structure)

Used protocol:

- CXL.io
- CXL.cache

Type-2

Devices which shows cache and memory to host



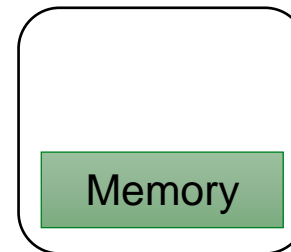
(e.g., GPGPU, FPGA which shows device internal memory to host)

Used protocol:

- CXL.io
- CXL.cache
- CXL.mem

Type-3

Memory expansion which connects CXL
Volatile memory and/or persistent memory

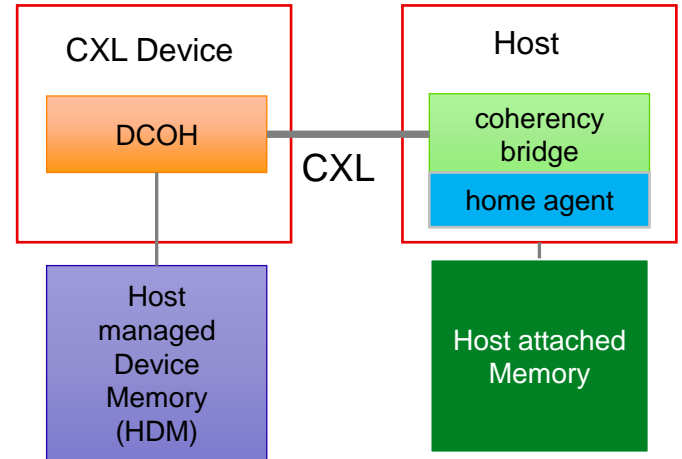


Used protocol:

- CXL.io
- CXL.mem

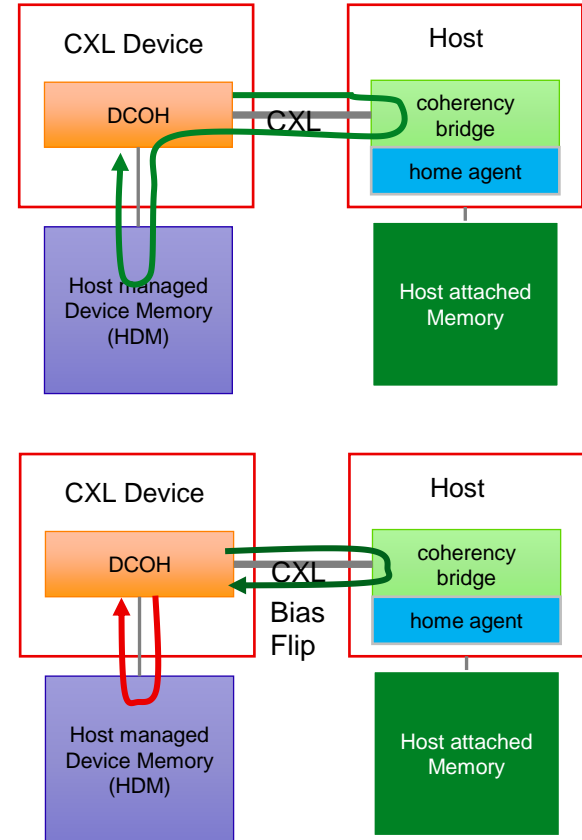
Cache and memory of CXL Type2 device

- CXL Type 2 devices manage their cache status by a **Device's Coherency Engine (DCOH)**
 - It is a component in the device
 - It must maintain status of cache of the device, and memory access
- Device memory which is included in the device and is shown to Host(CPU) is called **Host managed Device Memory(HDM)**



How to access from a Type2 device to HDM

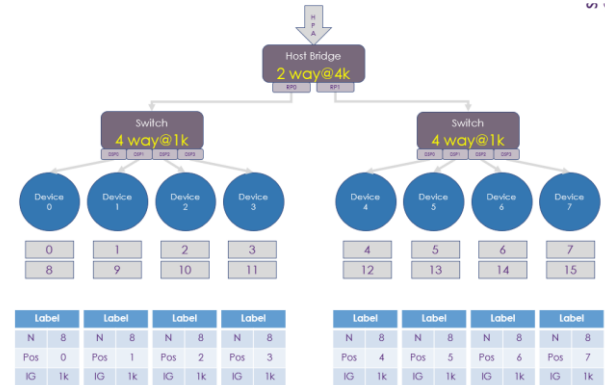
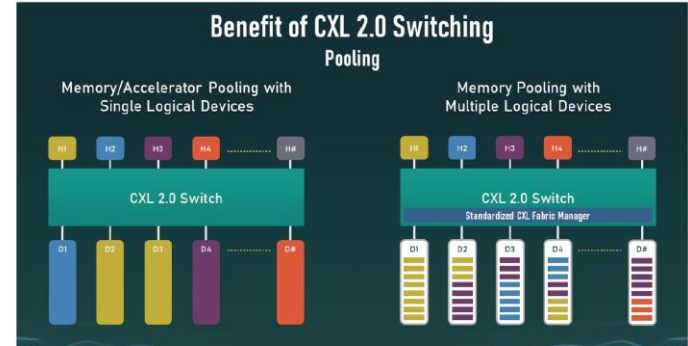
- A CXL device needs to select the following status to access its memory which is shown to Host CPU
 - Host Bias state (upper right)
 - Device needs to request to CPU to keep cache coherency before accessing device attached memory
 - Like the green arrow, it must send request to Host CPU once, then it can access device attached memory
 - Device Bias state (bottom right)
 - Device can access device-attached memory without consulting the Host's coherency engines
 - After Bias flip(green arrow), device can access ideal latency and bandwidth
- CXL 3.0 specification added another way (I'll talk it later)



Features of Type 3 memory device

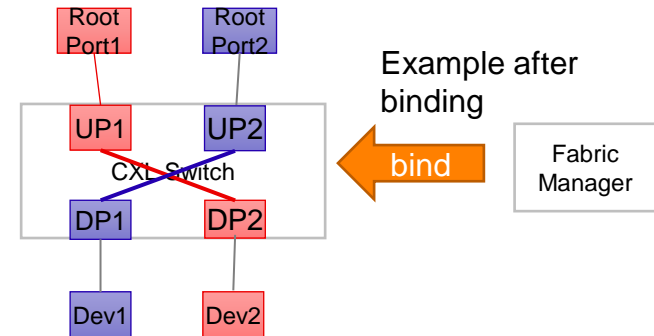
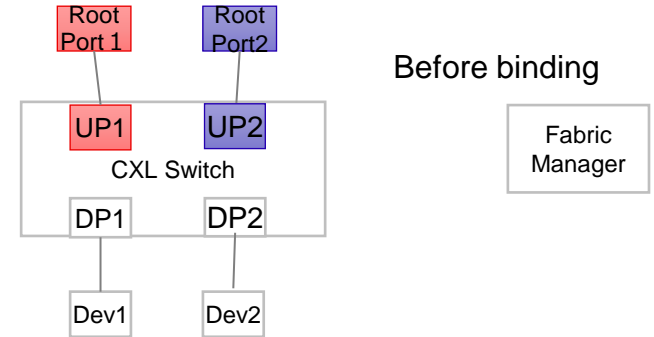
- The following configuration is available
 - You can configure devices as memory pool(Upper right) as the follows
 - Use one memory device to one memory region
 - Bind multiple devices to single memory region
 - Divide one device to multiple regions
 - Interleave is available (Bottom right)
 - Bottom right is an example of 8 way interleave by Host bridge and CXL Switch

CXL 2.0 Memory Pooling



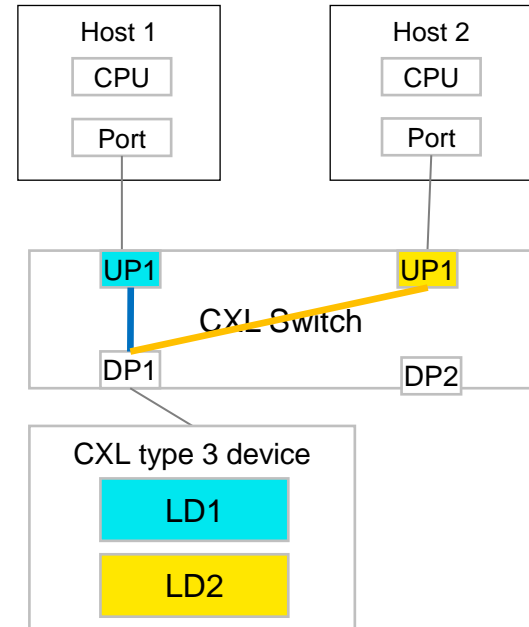
Binding of port of CXL Switch

- So far, Upstream port must be only one in a PCIe Switch
- A CXL Switch can have multiple upstream ports in it
- You can bind a down stream port to an upstream port dynamically
 - To configure the binding, a component which is called as **Fabric Manager** is necessary
 - Fabric manager can be implemented any style like the followings
 - Software which running on host machine
 - Embedded software running on a BMC like a server management software
 - Embedded firmware running on another CXL device
 - A state machine running within the CXL device itself



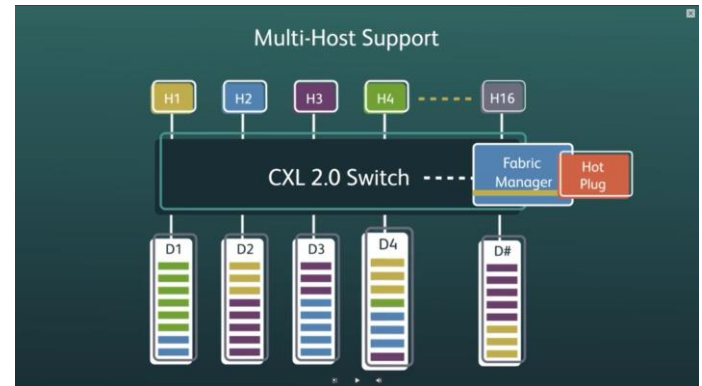
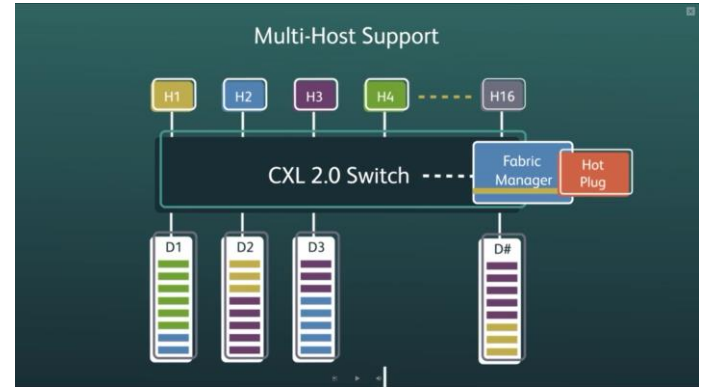
Divided Logical device

- Type 3 memory device can be divided into multiple regions as logical devices, and assigned to different hosts
- In the right figure, Type 3 device is divided to two logical devices
 - LD1(blue) and LD2(yellow) can be bound to different upstream port
 - These upstream ports may be connected different hosts
 - The Fabric Manager is responsible for dividing logical device and binding them to each port

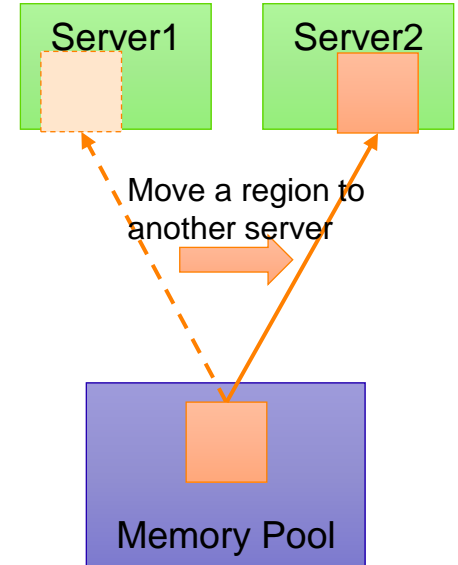


Hotplug is supported

- CXL 2.0 devices will be hot-pluggable like PCIe devices
 - It means CXL Type 3 memory devices will be hot-pluggable
 - Not only persistent memory, but also volatile memory will be hot-pluggable as a hardware specification
 - In past, Fujitsu made special servers which support volatile memory hotplug
 - Memory hotplug of Linux Kernel was developed for it at first
 - But many servers may support memory hotplug by CXL in future
 - Not only replacing a physical device, but you can add memory area which is hot-removed from another server
 - It will be important feature for memory pool



- Memory pool distributes a part of its regions to other servers as needed
 - Example of an old use case of banking system
 - Daylight : Give much memory to servers which process ATM transactions
 - Night : Give memory to other servers which process batch job like payroll transfer
 - So far, this feature is only possible by special server which support memory hotplug
 - Another option is to use virtual machines on the same host, but it is not possible to pass a memory area to other hosts
 - CXL makes it possible by establishing an open standard specification
- Failover
 - A server can take over regions previously used by another failed server
 - Not only memory, but also a GPGPU may be able to take over its processing in future



CXL 3.0 and 3.1

Specification updates

The list of new features of CXL3.0

- CXL 3.0 was released at 2022/Aug/1

- The right table is quoted from its [white paper](#)

- Personally notable features

- Fabric capabilities
- Memory Sharing
- Enhanced coherency

- Other things

- Twice speed than 2.0
 - It comes from PCIe 6.0 spec.
- Switching (Multi-level)
 - Allowing CXL switch hierarches
- Direct memory access for peer-to-peer
- etc.

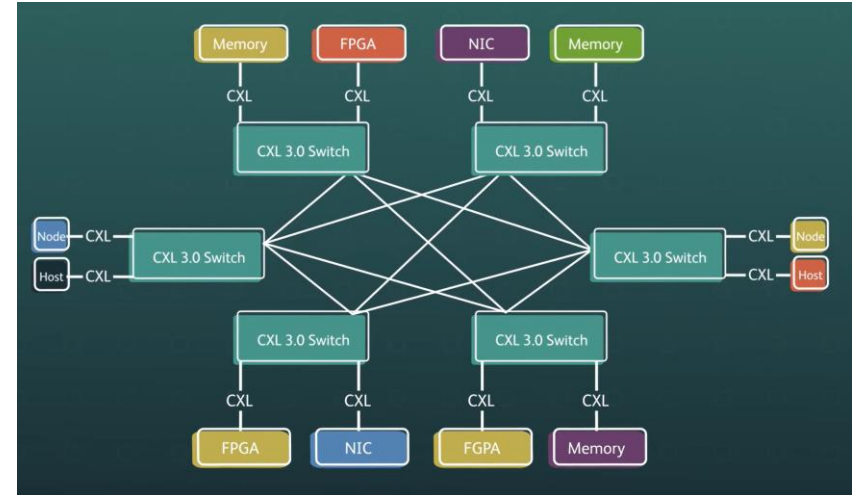
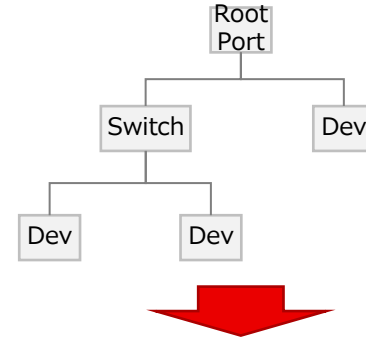
Features	CXL 1.0 /1.1	CXL 2.0	CXL 3.0
Release date	2019	2020	1H 2022
Max link rate	32GTs	32GTs	64GTs
Flit 68 byte (up to 32 GTs)	✓	✓	✓
Flit 256 byte (up to 64 GTs)			✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓
Global Persistent Flush		✓	✓
CXL IDE		✓	✓
Switching (Single-level)		✓	✓
Switching (Multi-level)			✓
Direct memory access for peer-to-peer			✓
Enhanced coherency (256 byte flit)			✓
Memory sharing (256 byte flit)			✓
Multiple Type 1/Type 2 devices per root port			✓
Fabric capabilities (256 byte flit)			✓

Figure 2: CXL Features over Generations

- Today, I'll talk about the three features which are notable I think

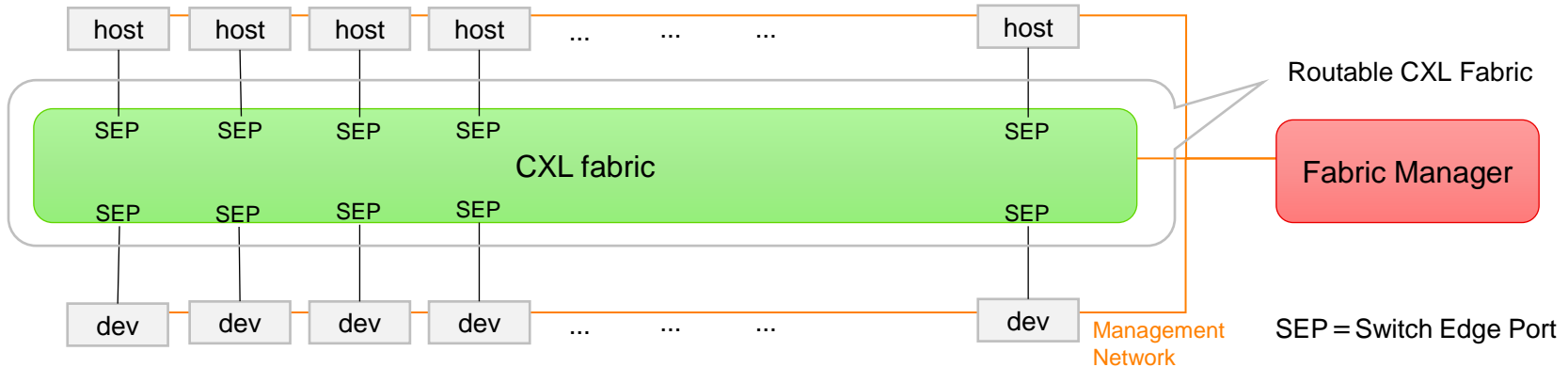
Fabric capability (1/2)

- Fabric connection is supported
 - The topology of connection was tree structure whose root was one root port until CXL 2.0
 - Even dynamic binding is available, tree structure is same with PCIe
 - CXL3.0 allows fabric connection via CXL Switch like the right figure
 - The # of maximum node is 4096
 - It can connect CXL devices with the shortest distance between servers
- This is the most notable new feature for me
 - It will be basis of the next generation of distributed computing



Fabric capability (2/2)

- Port Based Routing(PBR) is introduced
 - Messages in the fabric are sent with port ids of source and destination
 - Each id is 12bit for 4096 nodes
 - If a CXL switch support Port Based Routing, it is called PBR Switch
 - A Fabric manager needs to distribute ids to PBR Switches via “Management Network”
 - “Management Network” can be SMBus, I2C, I3C or Ethernet



- CXL 3.0 allows that a device can have cache coherency information
 - In CXL 2.0, only CPUs leads to maintain it
 - Device needs to ask CPUs to access its memory beforehand
 - In other words, CPUs and CXL device have **asymmetric** relationship for cache coherency
 - In CXL 3.0, the relationship between CPUs and device is **symmetric**
 - The DCOH of Device watches cache coherency information on CXL
 - In addition, the device can request CPUs to update their cache information if necessary
 - For this purpose, Back Invalidation Snoop(BISnp) channel is added to CXL.mem protocol

Enhanced Coherency(2/2)

- Specification describes a variety of access patterns and timings between a CPU and a device

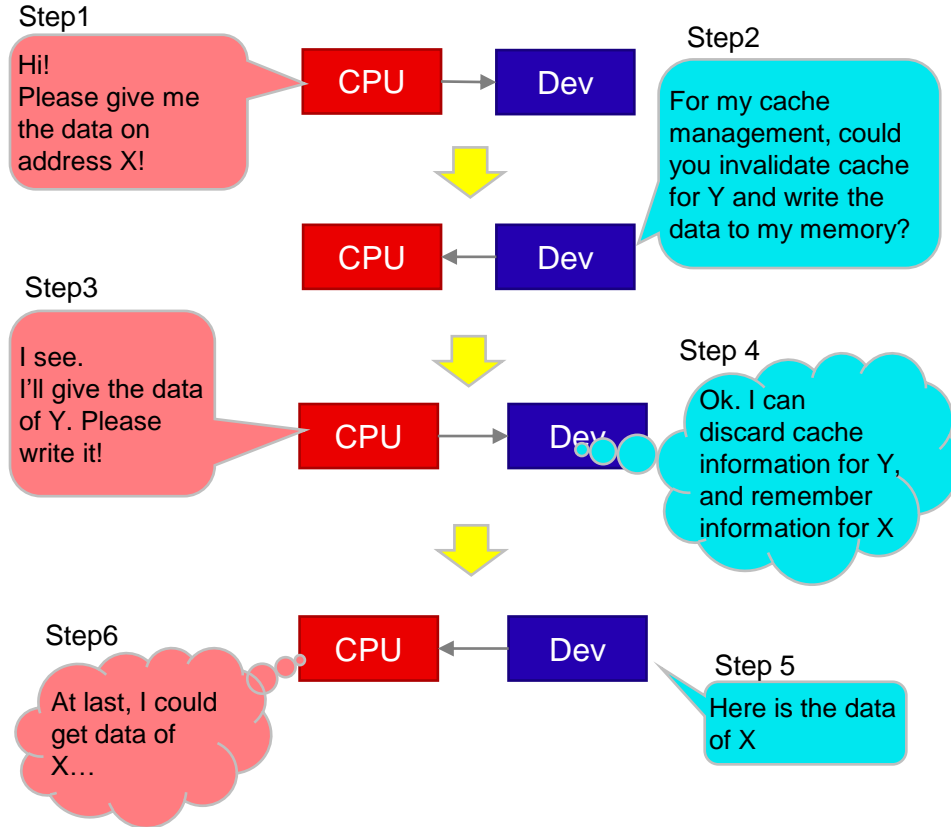
- The right figure is the simplified example of BISnp protocol

- Device can actively request CPU to change cache state and transfer data depending on the device state like right figure
- To confirm more correct sequence, please see the following section in the spec.

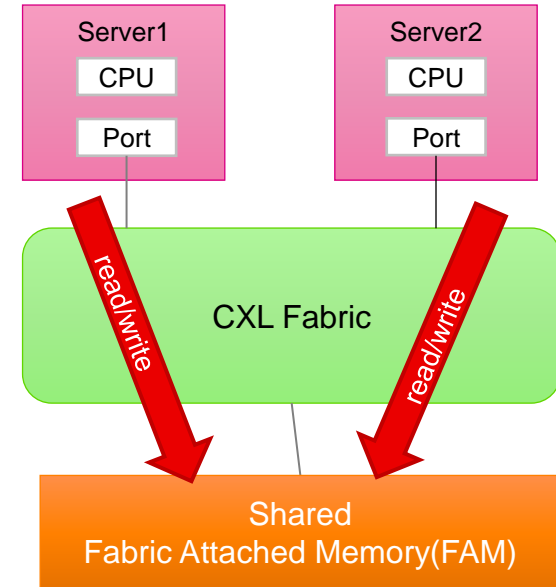
“3.5.1 Flows for Back-Invalidation Snoops on CXL.mem”

It describes various sequences, and you can understand how cache is managed actually

Please check it!



- Memory sharing between hosts(servers) is available
 - Each host can work together by shared memory
 - Fabric manager has role of configuration of which memory regions to share and how to share them
- There are two ways how to manage cache coherency
 - Multi-host hardware coherency
 - CXL device has a feature to manage cache coherency
 - When a CPU requests data write to a Device memory, the device need to coordinate cache information of other host CPUs
 - Software-managed coherency
 - Software need to manage cache coherency between hosts by itself
 - Even if the device does not have mechanism to coordinate cache coherency, this way is available
 - The actual mechanism by which software coordinates cache information is out the scope of the CXL specification



What is updated by CXL 3.1?

- Today, I will briefly outline some of updated features
 - Fabric Enhancements
 - More specifications have been added for the Fabric feature
 - Global Integrated memory(GIM)
 - It is used for enabling remote DMA and messaging across domains **via CXL Fabrics**
 - Dynamic routing
 - Message transfer can use different paths between source and destination ports dynamically
 - I suppose it seems a bit similar to the IP routing of TCP/IP
 - It is determined by congestion avoidance, traffic distribution across multiple links, or link connectivity changes
 - Security Enhancement
 - So far, CXL has supported CXL Integrity and Data Encryption (CXL IDE) feature
 - CXL 3.1 also supports Confidential Computing
 - CXL-TSP is defined
 - TSP = TEE (Trusted Execution Environment) **S**ecurity **P**rotocol
 - It defines mechanisms for allowing VM guests to execute within a trusted boundary on direct attached CXL memory

The status of current Linux for CXL

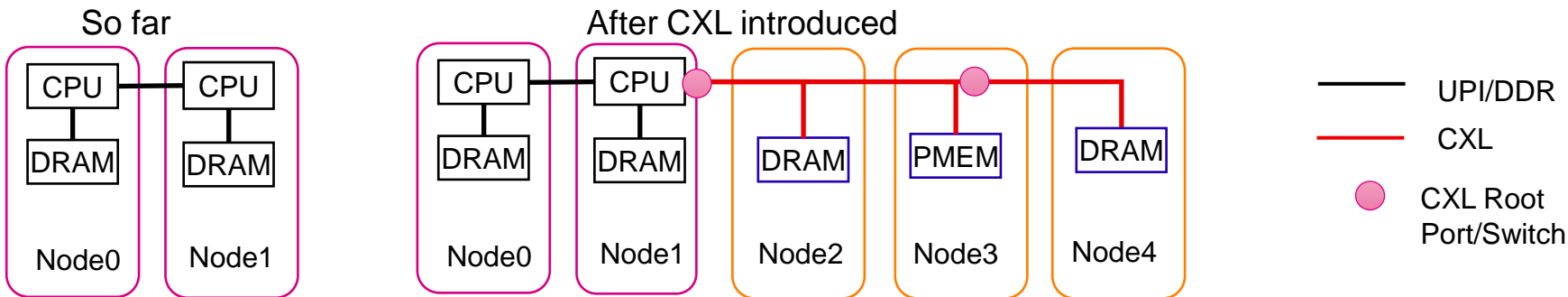
Summary of current status

- Basic implementation of CXL memory driver and commands has been developed
 - The driver can detect CXL memory devices
 - You can configure memory region, interleave by cxl command
 - The repository of cxl command is same with ndctl which is the command for persistent memory (<https://github.com/pmem/ndctl>)
- A solution for the “Memory tiering issue” was developed
 - CXL memory makes an environment which is called as “Memory tiering” due to variety of access latency
 - Since Linux memory management system did not consider it, new feature was developed
- There are some difficult issues for CXL memory hotplug/memory pool feature yet
 - Even CXL allows device hotplug feature as a hardware specification, Linux has some issues for CXL memory hotplug

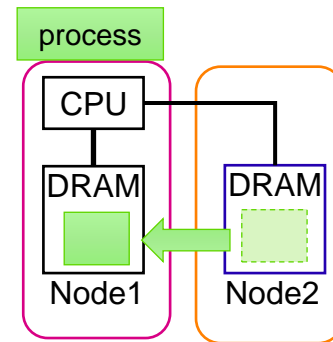
Today I'll talk about the latter two topics.

Memory Tiering(1/3)

- CXL memory has a difference of access latency compared to DDR memory
 - CXL Persistent memory will be slower than CXL DRAM memory
 - Accessing over a CXL switch is slower than direct accessing
 - As a result, memory access latency becomes “Tiering”
 - (The nearest DRAM from CPU)>(DRAM on another node)> (CXL DRAM)> (CXL PMEM)>....
- For this problem, CXL memory region is treated as a CPU less NUMA Node
 - Since Linux NUMA implementation considers for difference of memory latency, it is also suitable for CXL memory
 - There is no CPU in the CXL memory device, so the NUMA node of CXL memory becomes CPU less



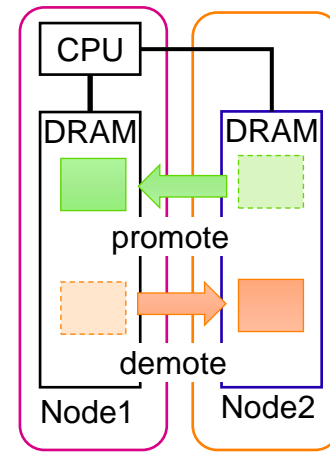
- In past, Linux memory management system did not have enough consideration for CPU less NUMA node
 - Current Linux NUMA balancing policy is to use nearest memory from CPUs
 - It allocates memory on the same node with the CPU which process executes if possible
 - If "Auto NUMA balancing" is on, contents of memory areas on a far node are moved to the node where the process is running
 - Since a CXL memory node does not have CPUs, process can not execute on the CXL memory node
 - As a result, CXL memory may not be utilized as expected even if NUMA balancing is used



Intel developed new feature to solve this problem

● Demotion/Promotion

- Instead of Swap Out and Swap In, kernel migrates cold page to CPU Less Node (demotion), and it migrates hot page to the nearest NUMA node from CPU(promotion)
 - So far, when a page is swapped out, CPUs cannot access its data until Swap In
 - However, even if a page is demoted, **CPUs can still access it**
 - Kernel decides which pages should be demoted in the page reclaim procedure
 - When a page is accessed by threshold time, kernel promotes it
 - The default threshold is one second, but the kernel automatically adjusts it based on the amount of promotion candidates and its bandwidth
 - This first work was completed in kernel 6.1 once
- In addition, the community continues to enhance the algorithm for selecting the demotion target
 - So far, kernel has only used old specification (ACPI SLIT) which provides only ratios against the nearest memory latency
 - Since ACPI HMAT can provide detail performance data, they are developing to use it
 - CXL CDAT may be used for it in the future



- CXL Memory hot-remove/memory pool has three big issues
 1. More software components are necessary for memory pool
 2. The CXL specification itself causes difficulty in hot-removing a CXL memory device
 3. **Not only specification, but there are many obstacles for memory hotremove in Linux**
- Unfortunately, I have not enough time to talk all of them today
- So, I will talk about **the last issue**

“What is obstacles for memory hotremove?”

- Please check appendix of my presentation about the other issues
 - You can see my presentation at Speaker Deck
- In addition, I would like to recommend that you read [my discussion at community ML](#) if you have more concern about these issues

Memory migration feature

- To talk the problem, I need to introduce how memory migration works

- To remove memory dynamically, contents on removing memory must be migrated to another place
- Kernel moves the contents of memory from removing memory to another memory without changing virtual address for it (right figure)

➔ **Memory Migration**

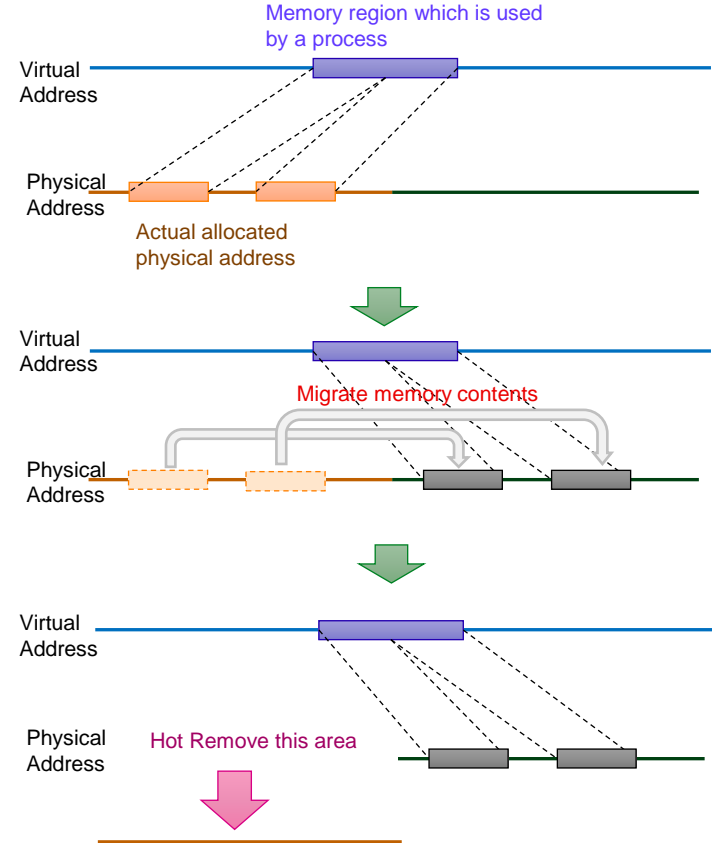
- Note:

- Basically, this can work for user process memory, but it cannot work for memory used by the kernel or drivers
- Because its virtual address must be changed when the physical address is changed

$$\text{Kernel Virtual address} = \text{Physical address} + \text{Fixed value}$$

- Unfortunately, even if the memory is used by user process, there are cases that memory migration can not work

You can not hot-remove such area



- Long term pinned pages are one of the big obstacle for memory migration

- RDMA feature (like InfiniBand) pins pages of user's process to transfer data from/to the pages without mediation by kernel
 - Kernel can not migrate "Pinned pages", because they may be under data transfer by the device
- I guess that DPDK or any fast performance devices/features may have same problem, and such kind of features will increase
- VM guest also tends to pin pages to skip kernel/hypervisor for performance improvement
- I think there is an ambivalent requirements like the followings

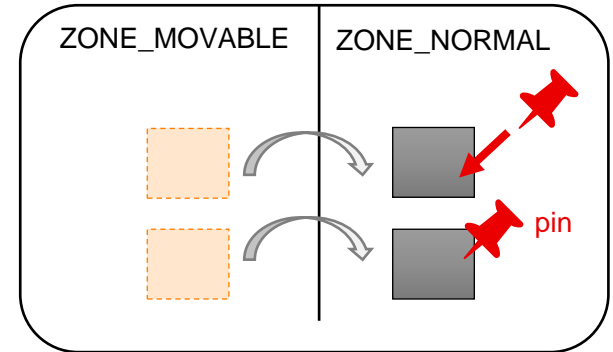
There are many things which want to **pin pages** and **skip kernel** to make better performance

V.S.

Kernel has the responsibility of any resource management in OS, and memory hotplug is one of them

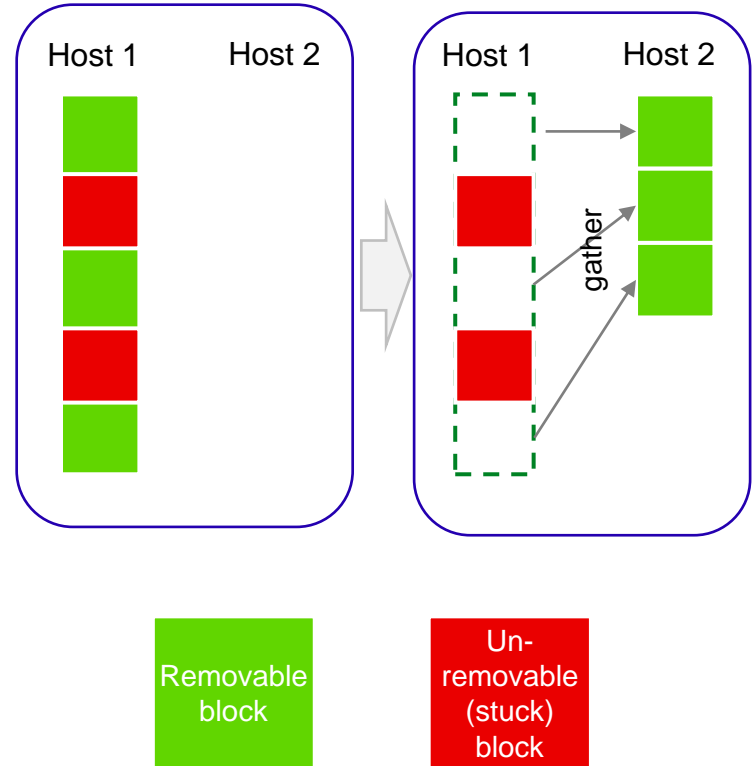
- The difficulty of improving CPU performance causes increasing the left side requirements
- Though kernel needs to manage the total balance of the system, it cannot be achieved by bypassing it
- I believe the root cause is a lack of communication between such feature and Linux kernel

- Before pinning the memory areas on removable memory, the kernel migrates contents of the areas to unremovable memory like DDR memory
 - The current kernel can create ZONE_MOVABLE areas based on user's configuration
 - ZONE_MOVABLE was created to ensure that removable memory is not used by the kernel or drivers
 - Therefore, it is beneficial for the CXL memory pool to allow hot-remove
 - To configure ZONE_MOVABLE, please refer to the memory-hotplug document in the kernel source code (Documentation/admin-guide/mm/memory-hotplug.rst)
 - If FOLL_LONGTERM flag is specified for the area of the data transfer target, the Linux kernel migrates them from ZONE_MOVABLE to another suitable place before pinning pages and data transfer (right figure)
 - This is reasonable solution for now
- But it may not be the final solution
 - If the amount of DDR memory is relatively too small compared to CXL memory, it may not be enough for pinning areas
 - VM guests also tend to pin pages of the hypervisor
 - If a large number of VM guests are executed, CXL memory may be not effective due to DDR memory shortage



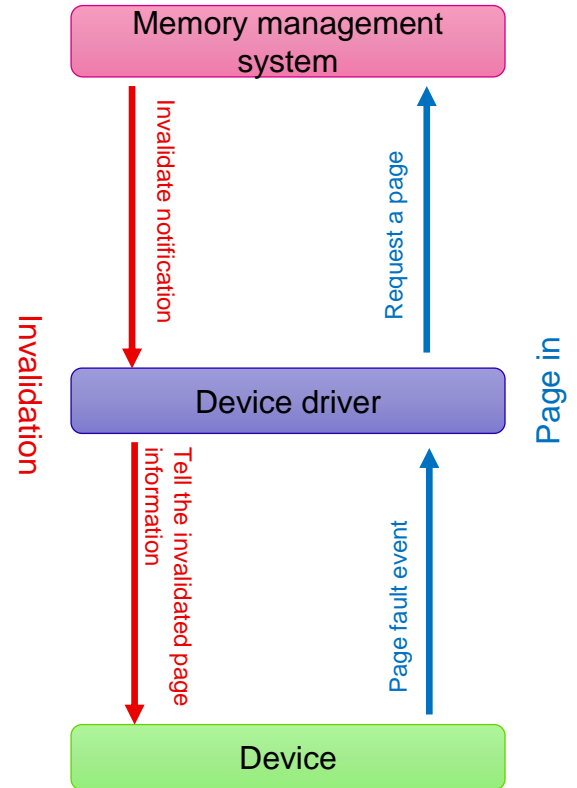
New approach of CXL specification

- “Dynamic Capacity Device” is introduced in CXL 3.0 specification
 - It allows that you can gather “removable” small memory blocks until required capacity rather than trying to remove stuck memory blocks
 - Then, they can be transferred to another host
 - “Memory pool” will be available by this specification
 - However, I suppose it may cause other problems
 - A CXL memory device will have mixture of the following memory blocks
 - Removable memory, Un-removable memory, Used by another host, Interleaved with other memory devices, and Shared by other hosts.....
 - It may be difficult to manage of the CXL memory devices
 - E.g., replace of the device




One of my idea for future

- “On Demand Paging(ODP)” may solve this issue
 - ODP is a way a device can transfer data by RDMA without pinning pages
 - It is **communicating** between a device and Linux kernel
 - When the kernel is going to invalidate a page for swap out or memory migration, it can notify the event to the driver and the device
 - When the device needs to access the invalidated page again, hardware asks the driver to execute procedure like page fault, and can restart data transfer
 - Currently, only NVIDIA(Mellanox) network cards support it yet
 - **However, I hope more hardware vendors will support it**
 - To understand ODP, I recommend the followings
 - The Mellanox [presentation](#) and their [paper](#) are very helpful to understand ODP
 - Since we are developing an [ODP support patch set for SoftRoCE](#), you can check and try it without any special hardware!



- I talked about CXL specification overview
- And talked about new specification of CXL3.0 and 3.1
- Current status of Linux Kernel development community



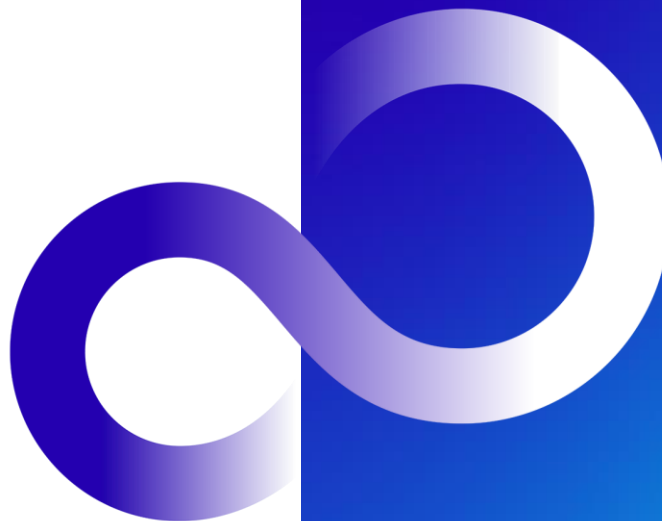
I hope that many vendor will release CXL devices and boost the market



I also hope that many people will develop features/drivers for CXL in Linux community



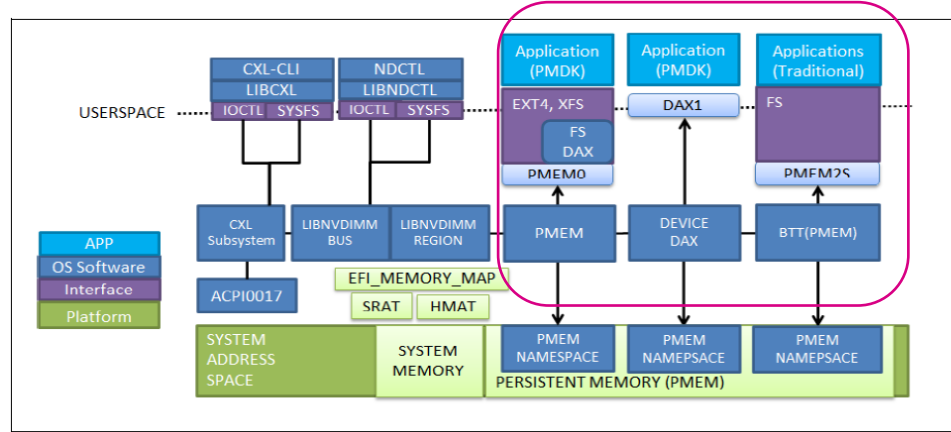
Thank you



Appendix

Interface of CXL type 3 persistent memory

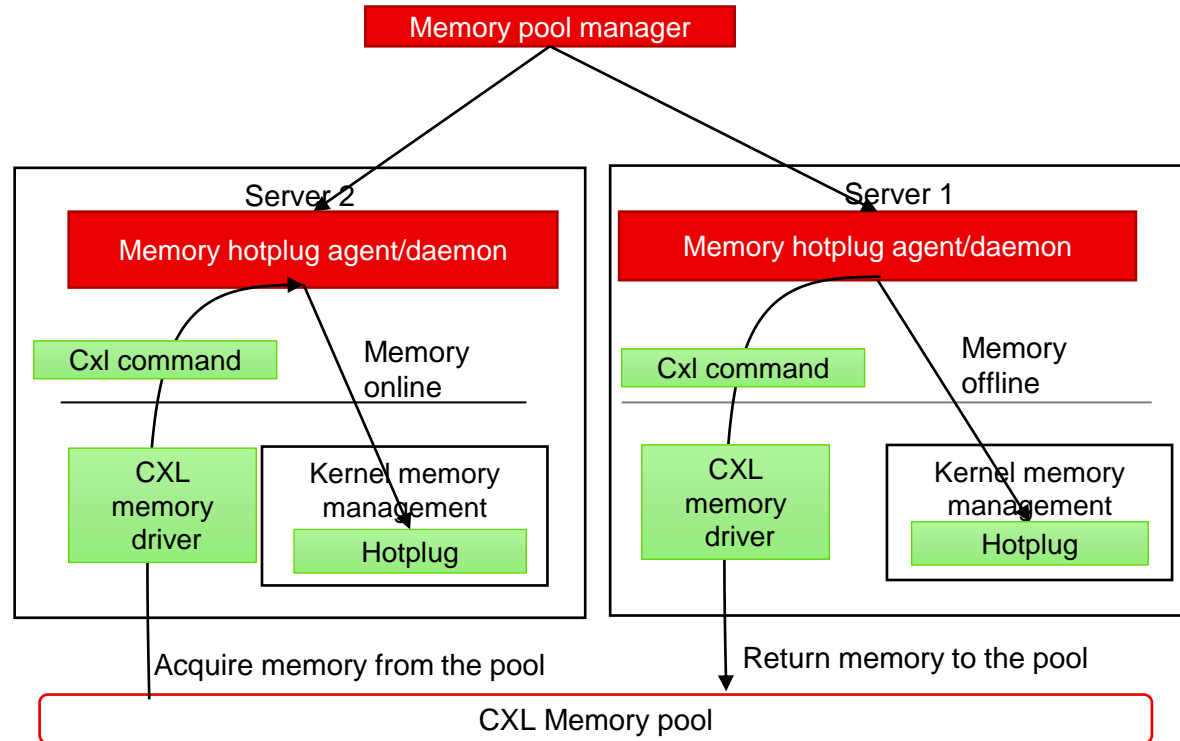
- CXL Type 3 device can be volatile memory or persistent memory
- Linux community needed to develop new drivers for cxl type 3 memory
- However, its interfaces of persistent memory DIMM are still available for CXL persistent memory device (Inside red frame in right figure)
 - Storage mode, Filesystem DAX, Device DAX



Software for Optane Pmem will be available on CXL Persistent memory device

More component is necessary for memory pool with Linux

- Two components are still necessary for memory pool, but there is no implementation
 1. Memory pool manager
 - It manages total memory pool, and orders each agent/demon to execute memory hotplug on each server
 2. Memory hotplug daemon
 - It watches memory usage in the server, gets request from the memory pool manager, and execute memory hotplug procedure
- I hope they should be vendor neutral, de fact standard OSS



CXL spec. is cause of difficulty of CXL memory device hotremove itself

- CXL specification itself is cause of obstacle of CXL memory **device** hotremove
 - Though CXL specification says that CXL device can be hotplugged as PCIe device, it may be too difficult
 - The removing memory devices may have other regions as multi logical device. It means that it affects many users of the device
 - The device may be interleaved with other devices
 - In addition, the device may be shared between multiple host

Users need to release all of the above configuration beforehand.
But, it is too difficult to manage

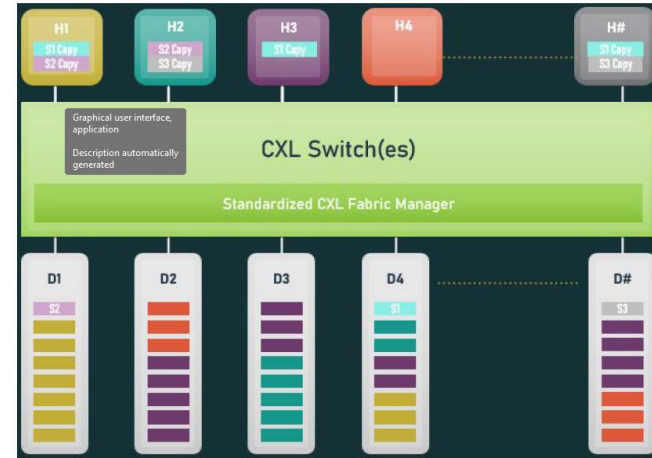


Figure 4: CXL 3.0 Memory Pooling and Sharing

- D4 is used by H1, H2, H3, and H#
- S1(Light blue) is shared by H1,

How do you replace when D4 is broken?

- CXL 3.1 specification
<https://www.computeexpresslink.org/download-the-specification>
- CXL 3.0 whitepaper https://www.computeexpresslink.org/files/ugd/0c1418_a8713008916044ae9604405d10a7773b.pdf
- CXL 3.1 whitepaper https://www.computeexpresslink.org/files/ugd/0c1418_6ede12bda4d34ffeb879c3700dde38f9.pdf
- lwn.net
<https://lwn.net/Articles/894598/> and <https://lwn.net/Articles/894626/>
- CXL* Type 3 Memory Device Software Guide
<https://cdrdv2.intel.com/v1/dl/getContent/643805?wapkw=CXL%20memory%20device%20sw%20guide>
- SDC 21 “Compute Express Link™2.0: A High-Performance Interconnect for Memory Pooling”
<https://www.snia.org/sites/default/files/SDC/2021/pdfs/SNIA-SDC21-Rudoff-CXL-Interconnect-Memory-Pooling.pdf>
- On Demand Paging for User-level Networking
https://openfabrics.org/images/eventpresos/workshops2013/2013_Workshop_Tues_0930_liss_odp.pdf
- Page Fault Support for Network Controllers
<https://courses.engr.illinois.edu/ece598ms/sp2018/papers/paper254.pdf>
- My discussion about memory hotplug in Linux Community
https://lore.kernel.org/linux-cxl/646e7f96f33e2_33fb3294c1@dwillia2-xfh.jf.intel.com.notmuch/T/#m8f3dd3d916600cce4ea088588add5501870c5241