EMUCXL:

an emulation and access library for CXL

Raja Gond
under the guidance of
Prof. Purushottam Kulkarni
rajagond@cse.iitb.ac.in

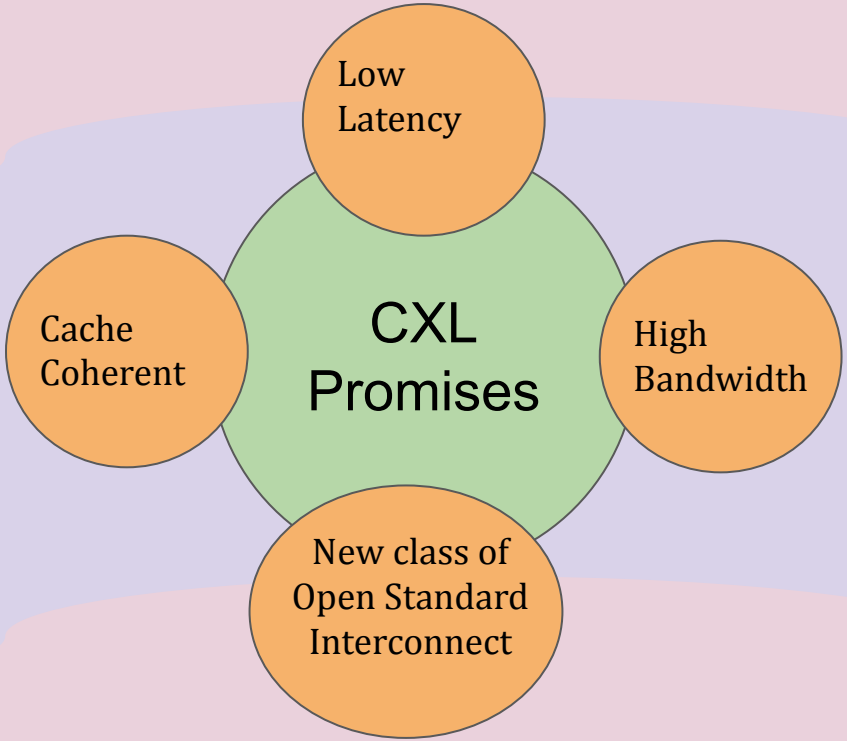
R&D2-Spring 2023

Outline

1. Introduction
2. Background and Motivation
3. CXL Emulation
4. Design of Emucxl
5. Discussion and Challenges
6. Future Work
7. Conclusion

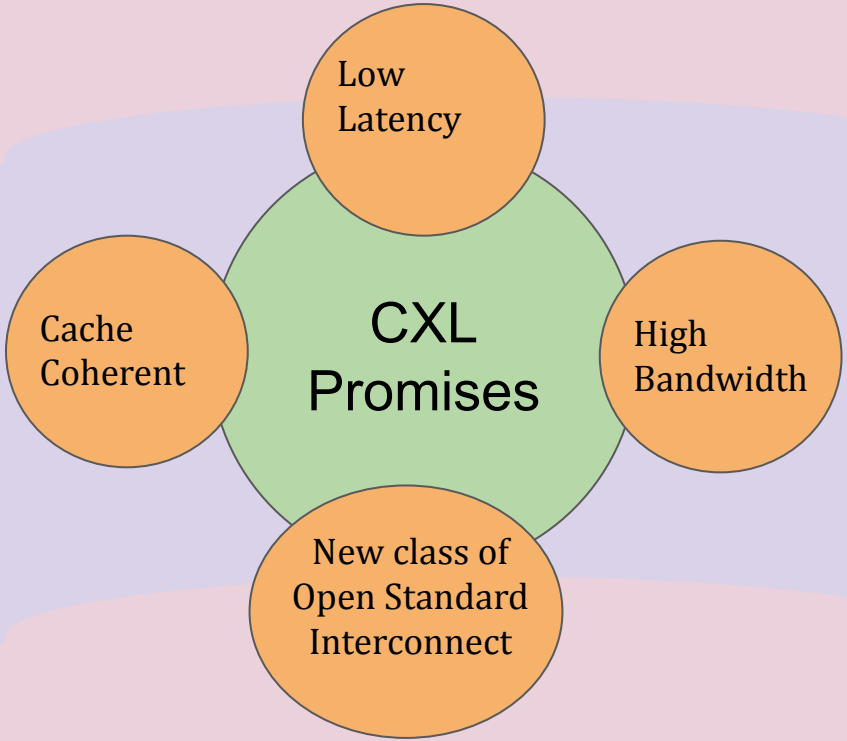
Introduction

- The emergence of Compute Express Link (CXL) in the interconnects market holds great promise for transforming the architecture of host-device interconnects.
- Interconnects: a pathway that allows different components within a computer system to communicate with each other.

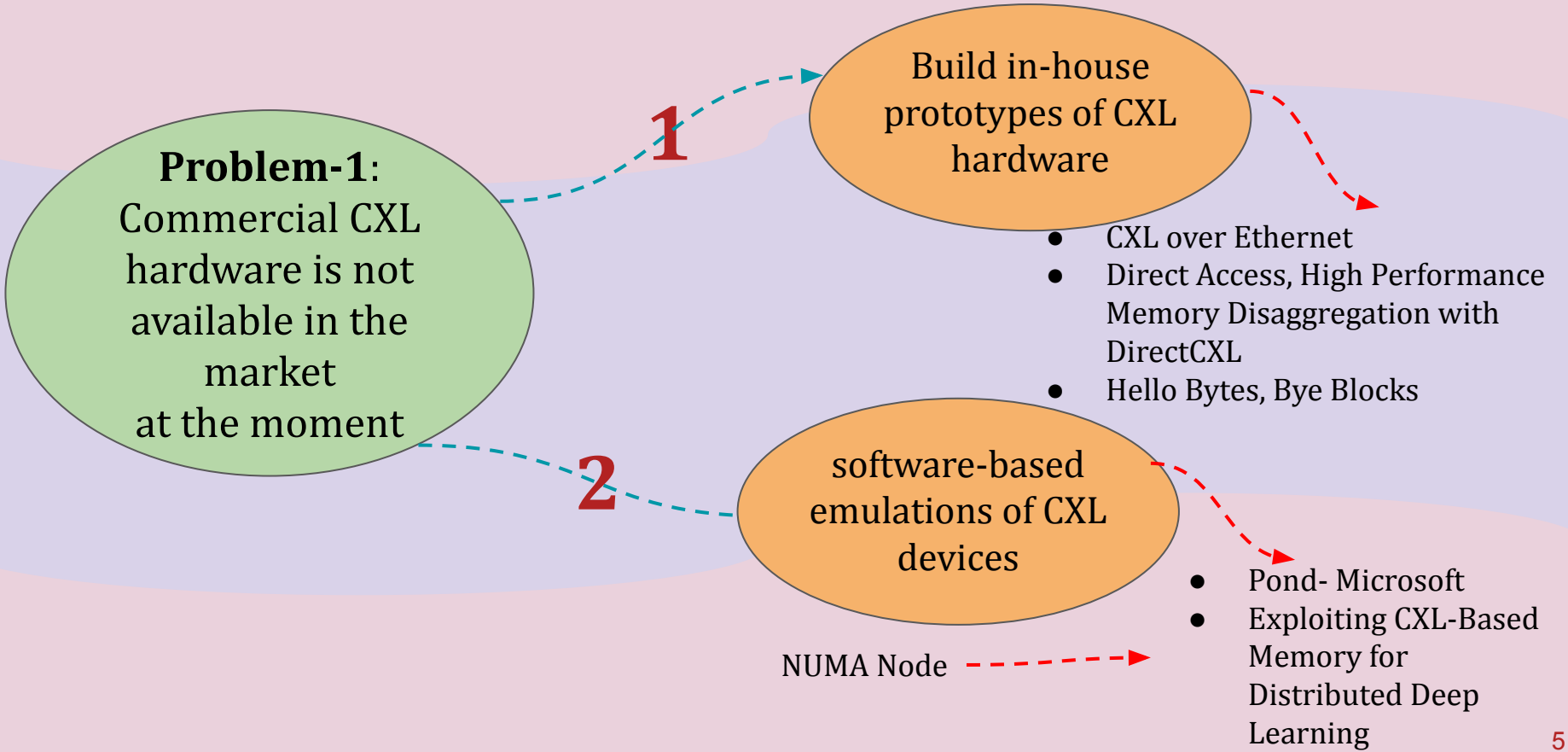


Introduction

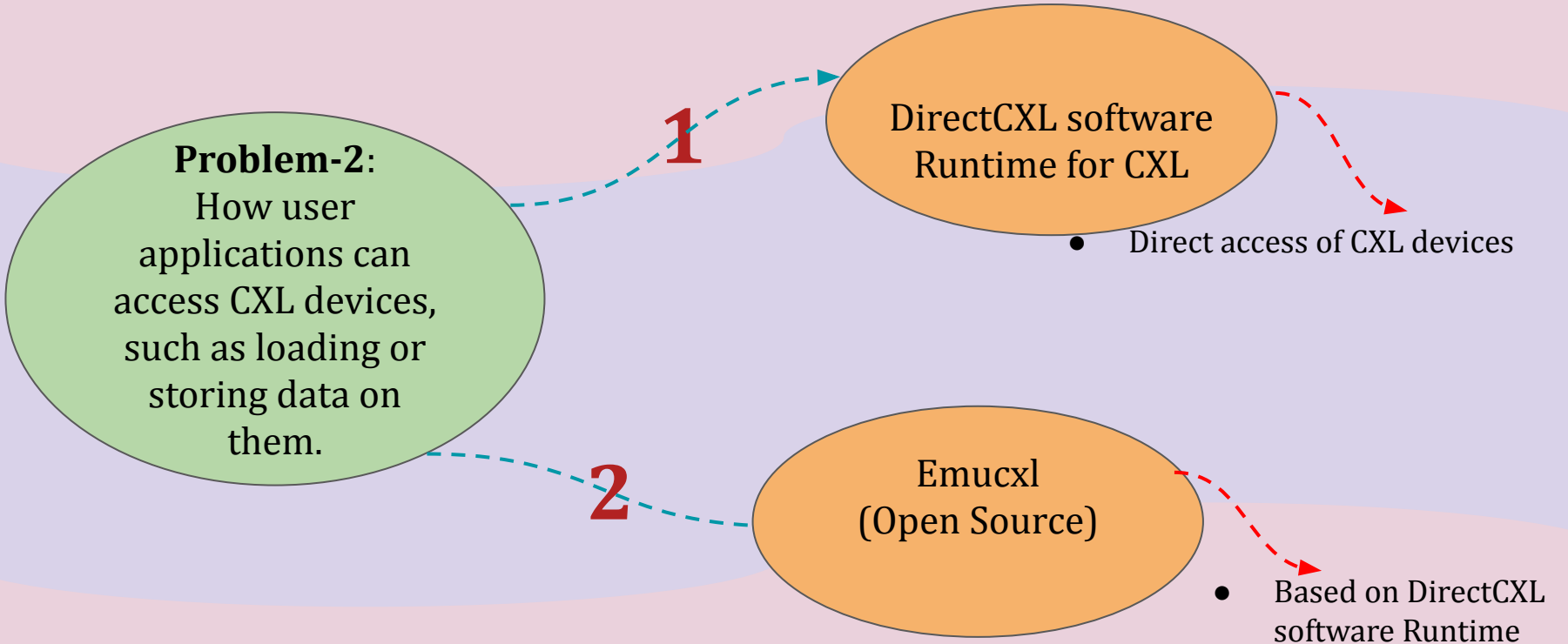
➔ **Emucxl** is a library that enables user applications to perform load and store operations on emulated CXL devices. It is build on the idea presented in the paper titled *Direct Access, High-Performance Memory Disaggregation with DirectCXL*



Background and Motivation



Background and Motivation



CXL-Emulation

CXL Compute Express Link™ + QEMU + Linux = YES

Possible but under active development

CXL hardware not yet available

Emulate CXL devices with **Qemu**

- Also require support from **Linux** Kernel
- Already in active development to support **CXL** devices.
- Load and Store aren't yet supported

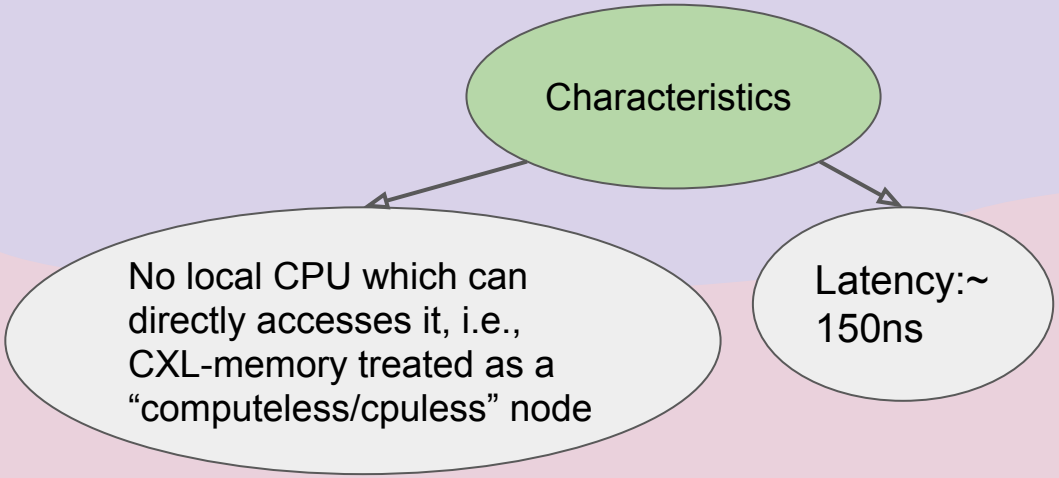
https://www.computeexpresslink.org/

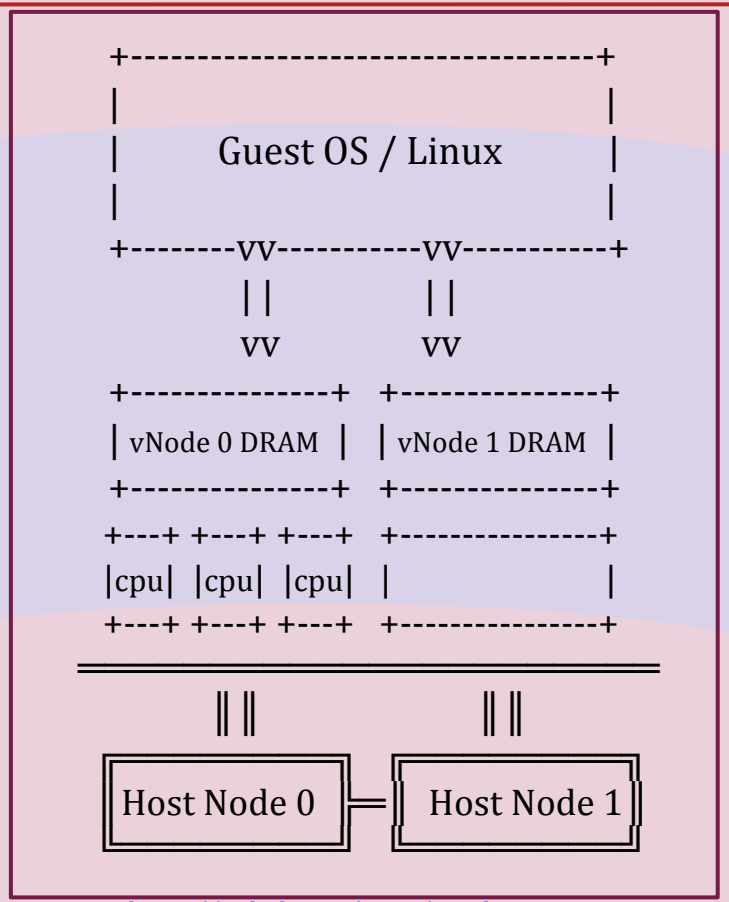
https://www.kernel.org/

CXL-Emulation

CXL Emulation on regular 2-socket (2S) server systems

- A paper called **Pond** by Microsoft emulate the following two characteristics of Compute Express Link (CXL) attached DRAM:



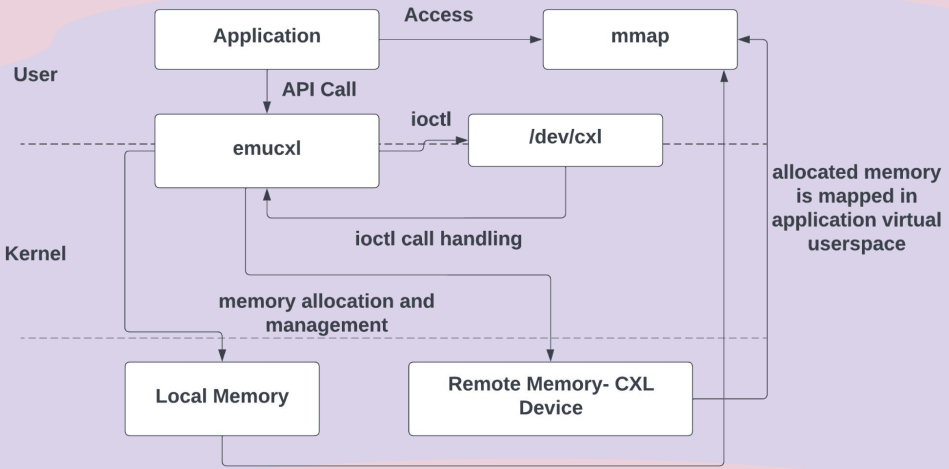


https://github.com/vtess/Pond

synerg@cse.iitb.ac.in

Design of EMUCXL

- We are using simplified approach and assume that only one user applications is using the emucxl.
- *kmalloc_node* or *vmalloc_node* to allocate memory on specific node
- *remap_pfn_range* to memory mapped the allocated memory to user virtual address space.



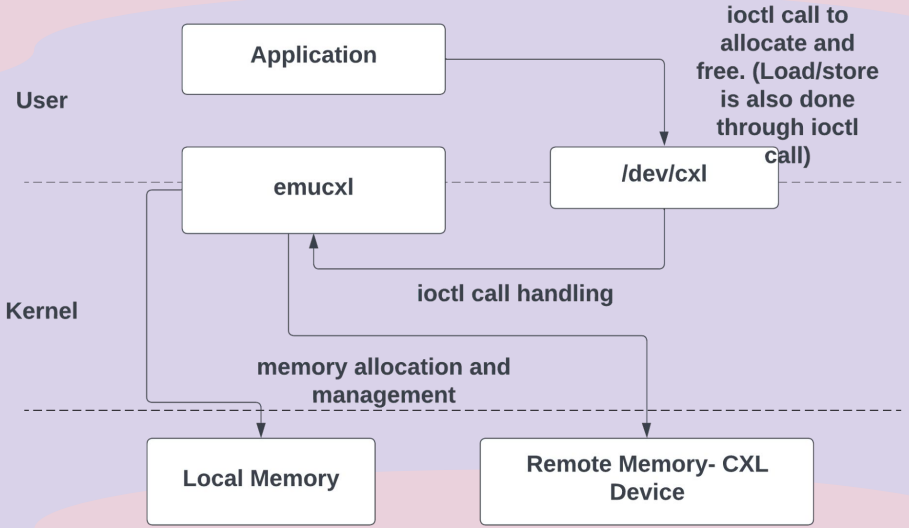
Design of EMUCXL

➔ Some APIs that was planned for implementation.

void emucxl_init ()	Set up the device file and other configurations required for Emucxl operation
void emucxl_exit ()	Close the device file
void* emucxl alloc (..., type tp , size t size)	Used to allocate memory either remotely or locally, depending on the type specified
bool emucxl free (..., void* address)	used to free allocated memory.
void* emucxl migrate (..., void * address , type tp , size t new)	used to transfer memory between local and remote locations, depending on the type specified.
bool emucxl resize (..., void* address , size t curr , size t new)	used to resize memory either in its original location or in the opposite location of the new size, if possible.

Design of EMUCXL

- The current implementation does not support any API calls or memory-mapped access.
- Challenges that comes along the way is discussed in next section



Discussion and Challenges

➔ NUMA-aware memory allocation

- vmalloc_node (size , NUMA_NODE) ;
- kmalloc_node (size , GFP_KERNEL , NUMA_NODE) ;
  // contiguous allocation
- kzalloc_node (size , GFP_KERNEL , NUMA_NODE) ;
  // contiguous allocation and initialise with zero
- vzalloc_node (size , NUMA_NODE) ; // initialise with zero

Discussion and Challenges

- ioctl calls
  - #define EMUCXL_ALLOCATE_MEMORY _IOWR (' e ' , 4 , emucxl_arg_t*)
    // ioctl command

- **copy_from_user** and **copy_to_user** API is used to transfer data.

Discussion and Challenges

- ➔ Challenge: Managing allocation for a combination of local and remote memory.
- ➔ Another challenge was allocating memory in kernel space with *kmalloc_node* and then mapping it to application userspace. Although I tried *remap_pfn_range*, but it resulted in program crashes. The mmap API is considered as a possible solution, but no numa-aware allocation is found.
- ➔ While *kmalloc_node* was able to allocate memory, *vmalloc_node* was not working in the EMUCXL setup.

synerg@cse.iitb.ac.in

Future Work

- ➔ API implementation
- ➔ Memory-mapped address space supports for user application
- ➔ Support for more than one user application accessing emucxl

Conclusion

The emergence of CXL represents a significant technological advancement that has the potential to transform data centre architectures. CXL specification is gaining wide traction in the industry due to the simplicity of implementing **low-latency caching** and memory semantics.

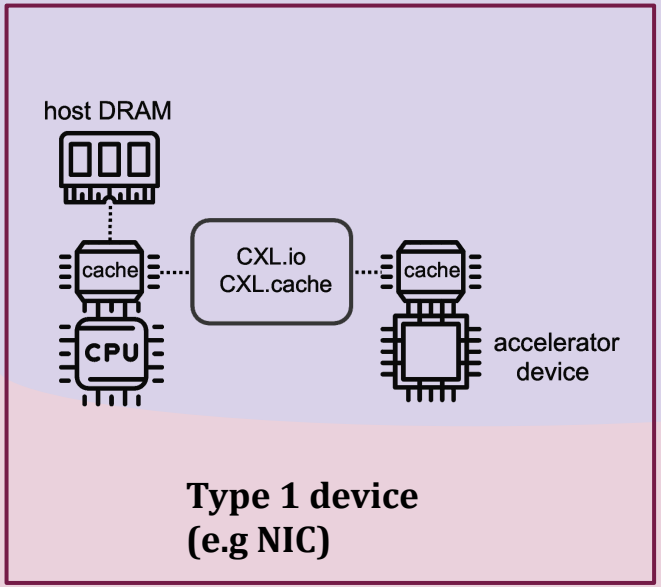
EMUCXL provides a simplified interface for user applications to get a feel of reading / writing on emulated cxl devices. The development of EMUCXL is a non-trivial task that required significant research, experimentation, and programming expertise.

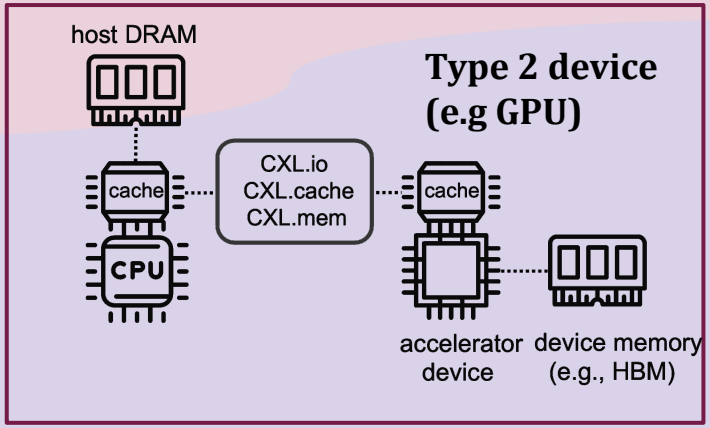
synerg@cse.iitb.ac.in

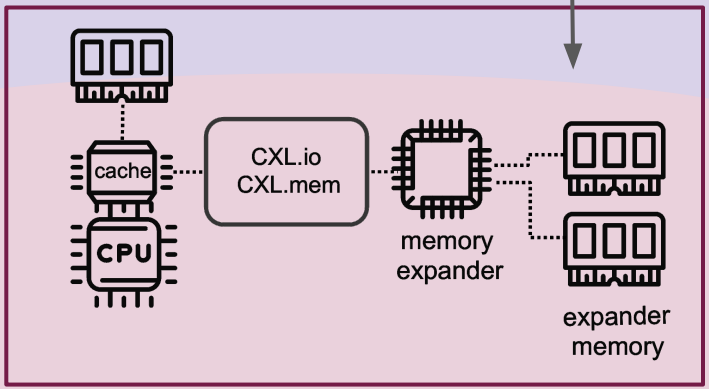
Thank you for listening!
Questions?

synerg@cse.iitb.ac.in

Type of CXL devices







https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10015193