[LSF/MM/BPF TOPIC]
SMDK inspired MM changes for CXL

Kyungsan Kim / Samsung Electronics

On behalf of SMDK* team

- We appreciate LSF/MM/BPF program committee for inviting and giving us the discussion opportunity.
- Also, we sincerely appreciate all the experts here for the advices, comments, interests on this topic.

SMDK*: Scalable Memory Development Kit, Samsung CXL SW for CXL Memory

Agenda Today

- Background
- CXL Requirement and SMDK Proposal

Background - SMDK

- CXL is a promising technology that leads to fundamental changes in computing architecture.

- As CXL DRAM provider, Samsung has developed both CXL DRAM HW and SW over last couple of years.
  To facilitate adoption and widespread of CXL DRAM, we have been developing a CXL SW development Kit, SMDK[1], since 2021 March working with industry and academic partners.
  Meantime, we **gained some kernel requirements from the works and customized SMDK kernel.**

- Also, CXL technology has been evolving thanks to many industry's efforts.
  As a result, CXL adoption stage is gradually moving forward **from basic enablement to real-world memory tiering usecases**. Around the stage, we would like to **discuss CXL requirements and introduce some of SMDK's kernel changes** to kernel maintainers/contributors here.

- But, please do not get us wrong. We want to explain our thoughts and approaches, but never force the approach.
  Personally, I majored OS and experienced kernel development since v2.4 around 2004.
  I respect kernel experts and strongly believe OS should be changed for a rationale reason and public use.

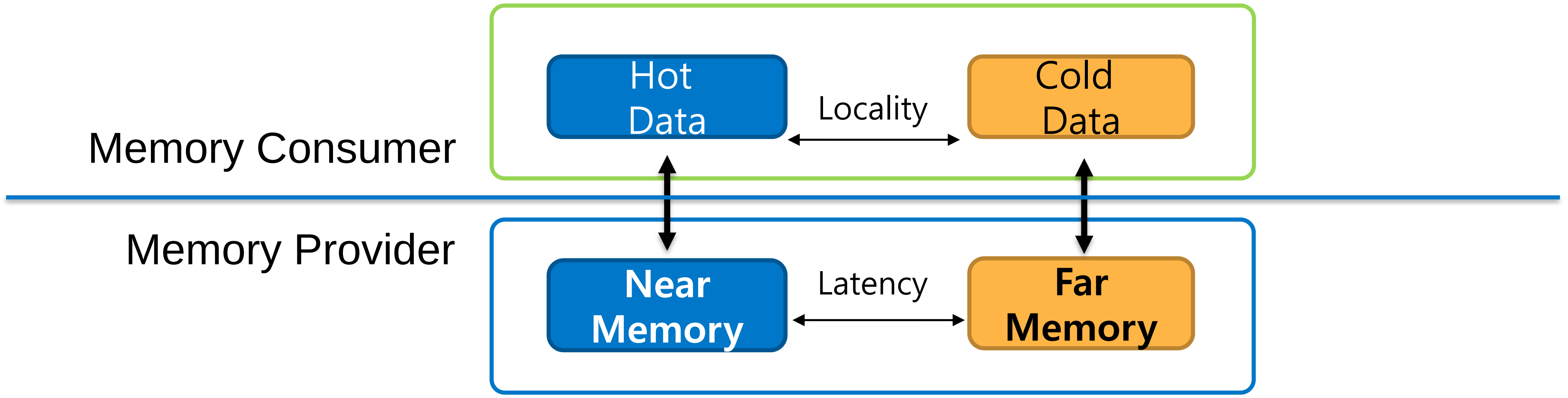
[1] SMDK: https://github.com/openMPDK/SMDK

Background - Memory Tiering Solution

- A system with CXL DRAM would consider a memory tiering solution.
  In terms of a memory tiering solution, it is typical that the solution attempts to
  locate hot data on near memory, and cold data on far memory as accurately as possible.
- The hot/coldness of data is determined by a memory consumer while near/far memory is determined by a memory provider. Thus, memory consumer needs **an identifier** to determine near/far memory.
- As memory vendor, SMDK put more weight on **near/far memory determinism** rather than hot/cold determinism, offering **memory tiering interfaces** for a memory consumer context at user/kernelspace.

- **The following 5 requirements and 2 proposals** are originated from the backgrounds.

Memory Tiering Abstraction



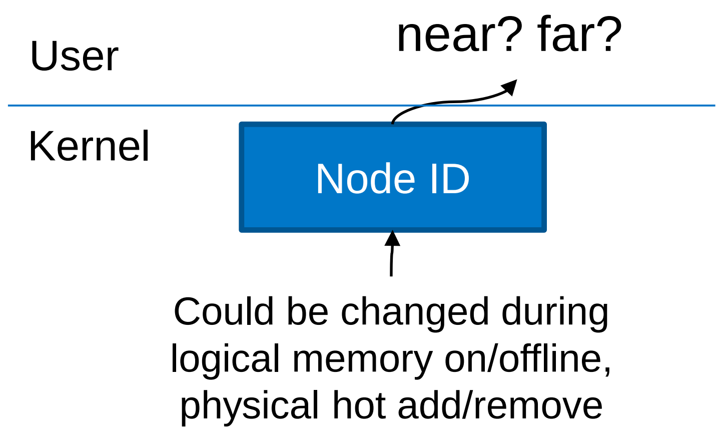
SAMSUNG

CXL Requirements (1)

- **1. CXL DRAM identifier (API and ABI)**

  Issue: a user/kernel context has to use the node id of a CXL memory-node to access CXL DRAM.

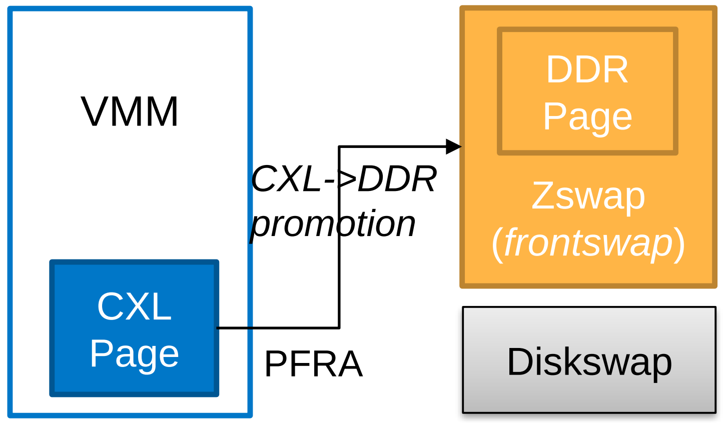
  Thought: Node id would be ephemeral information that can be changed. In addition, it does not present a near/far attribute of the node. A userspace and kernelspace memory tiering solution need API and/or ABI to identify near/far memory node.


User
near? far?
Kernel
Node ID
Could be changed during logical memory on/offline, physical hot add/remove

- **2. Prevention of unintended CXL page migration**

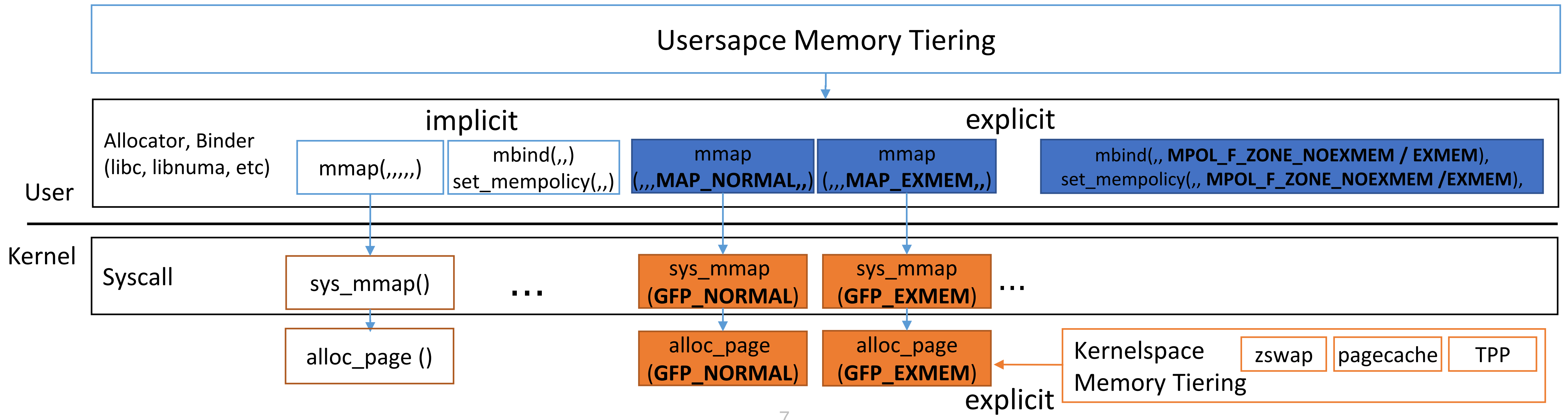
  Issue: In order to store swapped-out page on far memory(CXL DRAM), a page on near memory(DIMM DRAM) is allocated while zswap works.

  Thought: On the swap flow, a context that was employed a far memory should not be promoted to employ near memory accidentally.



LSF/MM – SMDK Proposal (1)

- We provide userspace/kernelspace programming interfaces to **explicitly** (de)allocate memory out of DIMM DRAM and CXL DRAM.
  - Syscall - mmap(), mbind(), set_mempolicy()
  - Kernelspace - alloc_page()
- Currently, only a userspace context is able to allocate CXL DRAM **implicitly**.
  - Kernelspace has to request CXL memory explicitly to avoid unpluggable condition by chance.



CXL Requirements (2)

- **3. CXL DRAM pluggability**

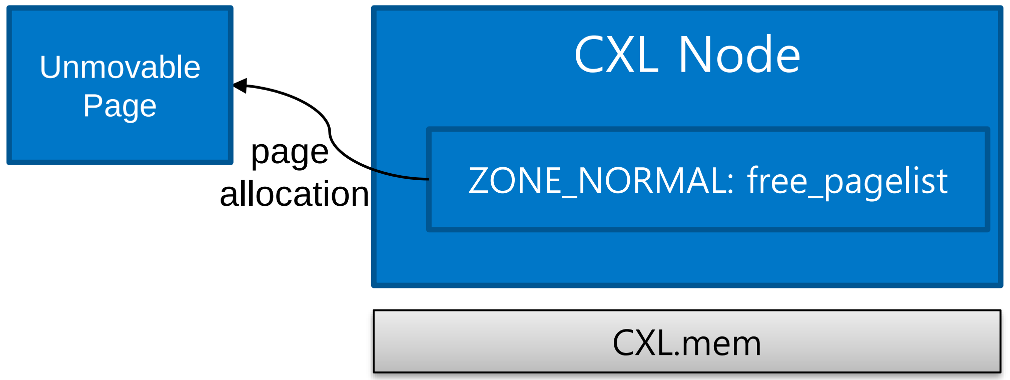
  Issue: a random unmovable allocation can make a CXL DRAM unpluggable.

  It happened out of kernelspace - pinning for metadata such as struct task_struct, page, zone, etc - or even rarely userspace - pinning for DMA buffer.

  By the way, we should separately think logical memory on/offline and physical memory add/remove for this issue.

  Thought: a CXL DRAM should be able to be used in a selective manner, pluggable or unpluggable.

  I apology for confusion while discussion. Don't get this wrong. Those are mutual-exclusive, so it cannot happen at the same time on a single CXL DRAM channel.



fail to offline/remove

CXL Requirements (2)

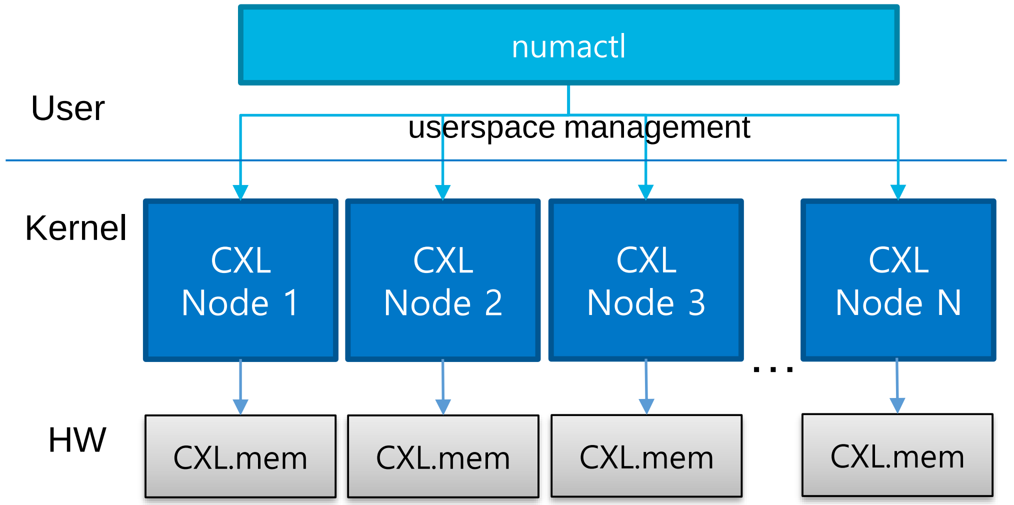
4. Too many CXL nodes appearing in userland

Issue: many CXL memory nodes would be appeared to userland along with development of a CXL capable server, switch, and fabric topology. Currently, a userland needs to be aware and manage the nodes using a 3rd party SW such as numactl and libnuma. e.g.) lead to aggregated bandwidth among the CXL nodes.

Thought: Kernel would provide an abstraction layer to deal with the node seamlessly.

Traditionally a node implies multiple memory channels from the same distance, so we thought that multiple CXL DRAMs can be appeared as a single node as well as separated nodes.

By the way, node is the largest management unit in MM. i.e.) Node - Zone – Page. Also, historically a new zone has been added to properly deal with a new different HW and SW algorithm. What if the management dimension for a single CXL DRAM would be smaller than node? or do we need a superset node?



CXL Requirements (2)

- **5. Flexible ways to use CXL DRAM to allow a variety of potential usecases**

  Issue : -

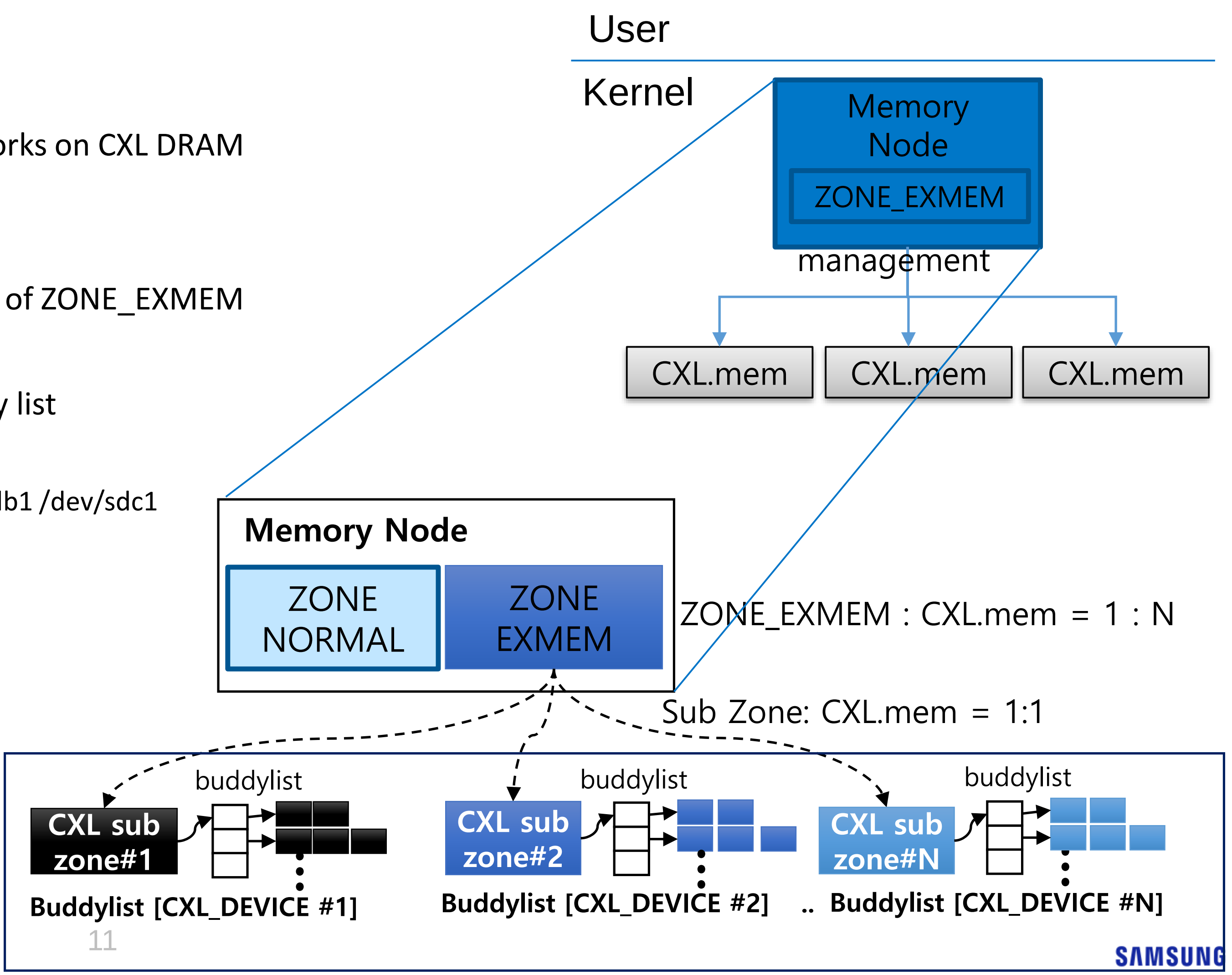
  Thought:

  – CXL, DDR/CXL channel interleaving(System/Switch FW) $\rightarrow$ +software interleaving

  – HDM grouping(System/Switch FW) $\rightarrow$ +after OS boot

LSF/MM – SMDK Proposal (2)

- A new zone, ZONE_EXMEM as **a separated logical management dimension** for physical CXL DRAM device.
- What ZONE_EXMEM concern?
- **1. Pluggability**
  - Not confine movable or unmovable attribute
  - It is the same with ZONE_NORMAL in that aspect, but it works on CXL DRAM
- **2. CXL Identifier** for
  - Beneath the Syscall and kernel allocator
  - MAP_EXMEM and GFP_EXMEM flag traverse free_pagelist of ZONE_EXMEM
- **3. Node Abstraction**
  - Node – Zone_EXMEM(1:CXL N) – Subzone(1:CXL 1) – Buddy list
  - Capacity/Bandwidth and Aggregation/Isolation
    - ✓ e.g.) mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1
    - ✓ e.g.) cxl group-add --target_node 1 --dev cxl1 cxl2
- **4. Zone level Algorithm** (for CXL)
  - Larger System RAM - Compaction, Reclaim watermark
  - Performance –Link Negotiation, QoS Throttling
  - Error handling - RAS, Switch/Fabric connection error
  - Sharing – Security, Permission
  - Async operation - Background (FW Update, Sanitize, etc)



SAMSUNG

Thank you !