SAMSUNG

[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.

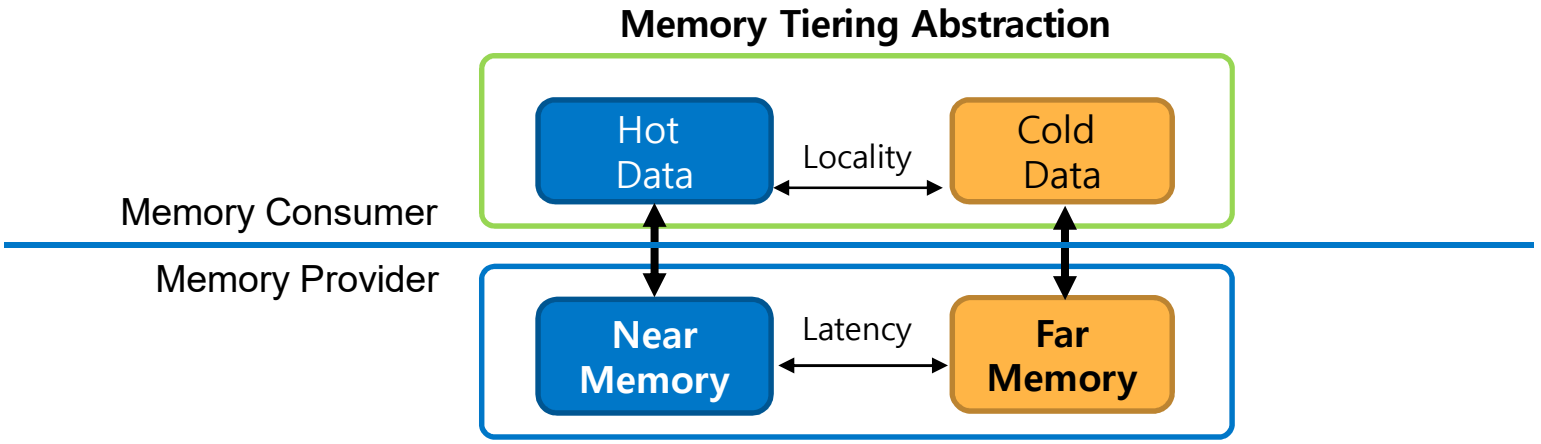
- But, please do not get us wrong. We want to explain our thoughts and approaches, but never force the approach.
  Personally, I majored OS and have experienced kernel development since v2.4 around 2004.
  I respect kernel experts and strongly believe OS should be changed for a rationale reason and public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



CXL Requirements (1)

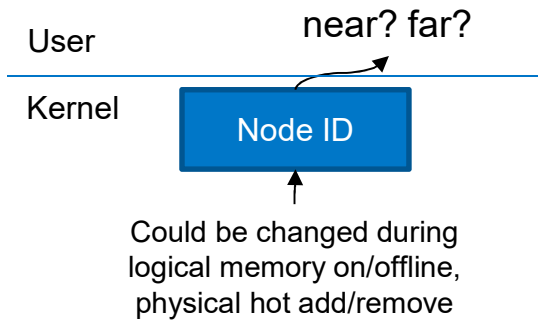
- **1. CXL DRAM identifier (API and ABI)**

  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.

  A 3rd party plugin such as libmemkind, libnuma can resolve it.

  However, using a plugin is not always preferred, so we propose a primitive API from kernel.

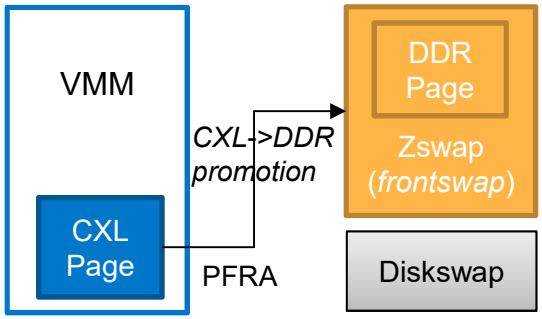


- **2. Prevention of unintended CXL page migration**

  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.

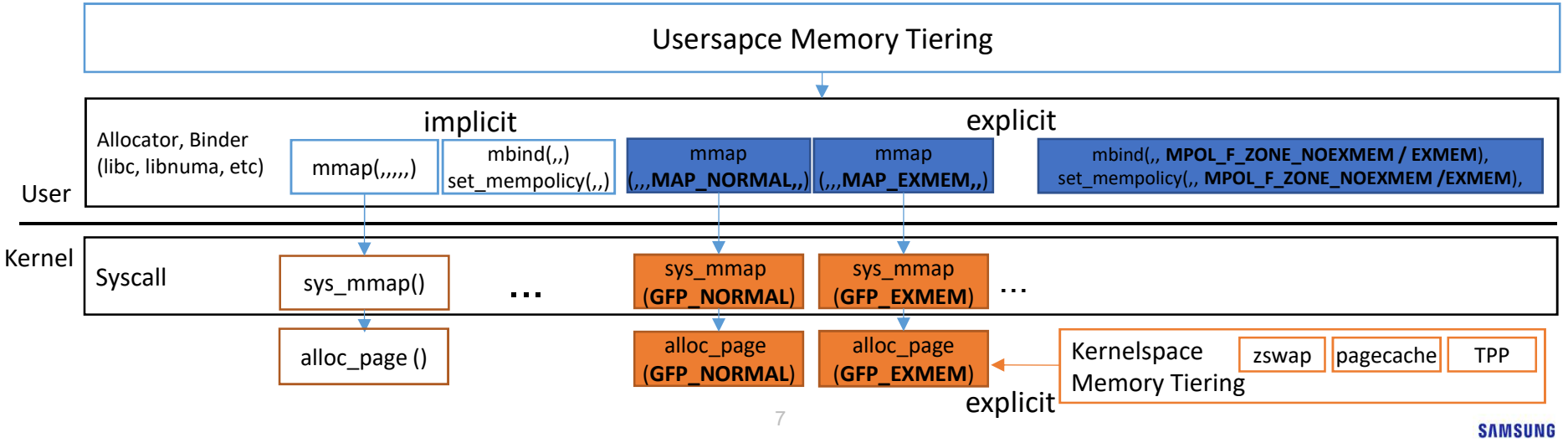
  It would happen other page migration flows as well based on node id.



SAMSUNG

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(MAP_NORMAL|MAP_EXMEM), mbind(), set_mempolicy()**
  - **Kernelspace - alloc_page(GFP_NORMAL|GFP_EXMEM)**
- 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.



SAMSUNG

CXL Requirements (2)

- **3. CXL DRAM pluggability**

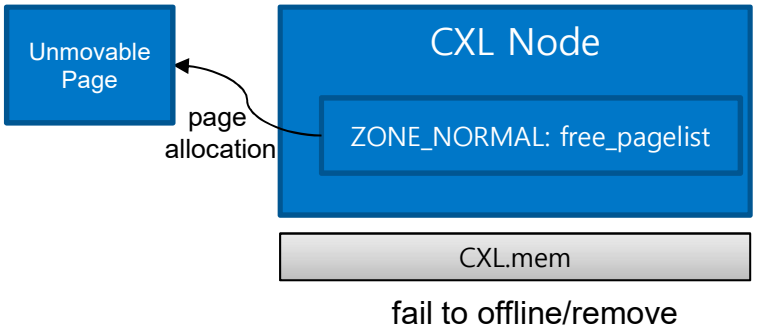
  Issue: a random unmovable allocation can make a CXL DRAM unpluggable.

  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.



CXL Requirements (2)

4. Too many CXL nodes appearing in user and kernelspace

Issue: many CXL memory nodes would be appeared to user and kernelspace along with development of a CXL capable server, switch, and fabric topology. Both need to be aware and manage the increased nodes.

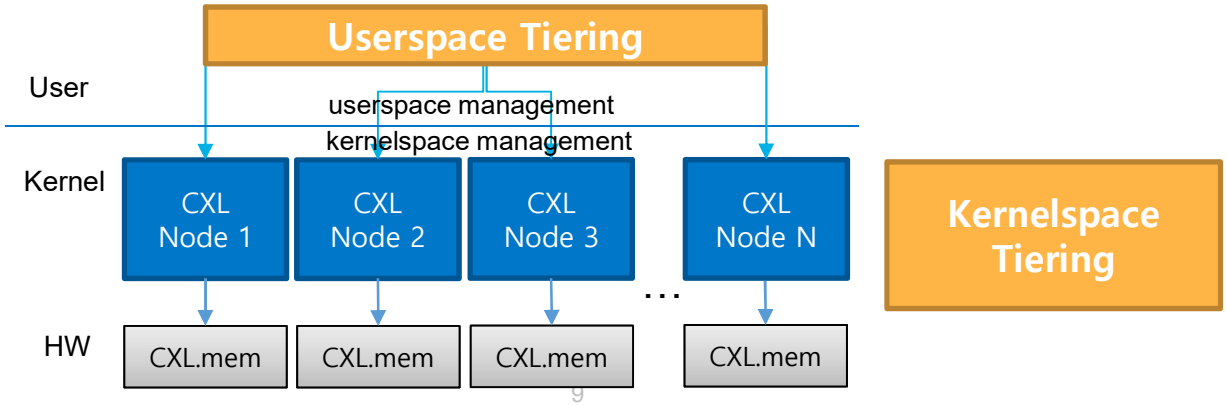
To be specific, a userland need to use a 3rd party SW such as numactl, a heap allocator.

A kernelspace tiering which is node-basis would be influenced as well such as Auto-numa, TPP.

Thought:

Kernel would provide an abstraction layer of nodes to deal with the increased node seamlessly.

Traditionally a node has implied multiple memory channels from the same distance.



CXL Requirements (2)

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

  Issue : No one mentioned yet. a CXL system being made is

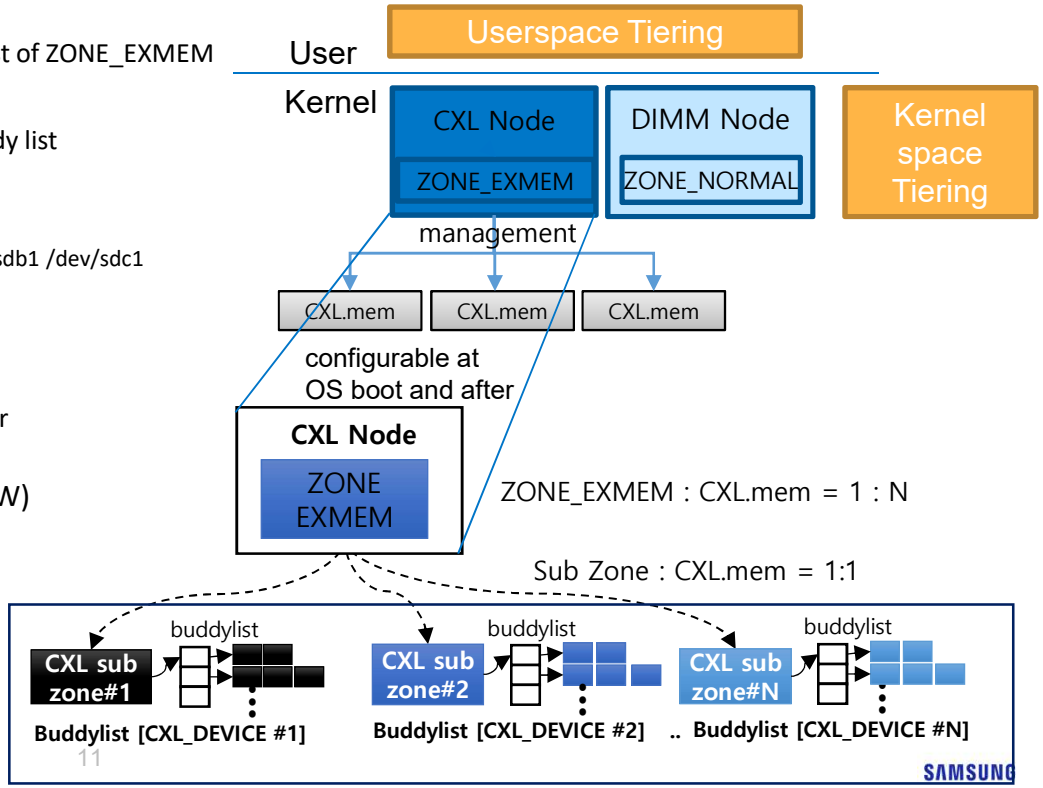
  - CXL.mem interleaving, DDR/CXL.mem channel interleaving by BIOS, Switch FW
  - CXL.mem grouping by BIOS, Switch FW

  Thought

  - 1.CXL Switch will be used at all times?
    - ✓ Direct attached → (Multi) Switched → Fabric connected, aiming composable datacenter ultimately.
    - ✓ IMHO those would be used in a mixed way for different purpose.
      e.g) Backplaine EDSFF(e3.s/e1.s) would be better than DIMM F/F for flexibility and maintenance aspect.
  - 2. BIOS setting is enough to support all real-world usecases?
  - So, what if allows the interleaving and grouping in software way after OS boot

LSF/MM – SMDK Proposal (2)

- What ZONE_EXMEM do?
- **1. CXL Identifier (Requirement 1,2)**
  - Beneath the Syscall and kernel allocator
  - MAP_EXMEM and GFP_EXMEM flag traverse free_pagelist of ZONE_EXMEM
- **2. Node Abstraction (Requirement 4,5)**
  - Node – Zone_EXMEM(1:CXL N) – Subzone(1:CXL 1) – Buddy list
  - Aggregation/Isolation of Capacity/Bandwidth
    - ✓ cxl group-add --target_node 1 --dev cxl1 cxl2
    - ✓ e.g.) mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1
- **3. Pluggability (Requirement 3)**
  - Not confine movable or unmovable attribute
    - ✓ The same of ZONE_NORMAL, but works on CXL DRAM
  - ZONE_MOVABLE(**ZONE_PREFERRED_MOVABLE, David**) or ZONE_NORMAL works for CXL DRAM
- **4. Zone level Algorithm** (would influence MM due to CXL HW)
  - Performance - Link Negotiation, QoS Throttling
  - Error handling - RAS, Switch/Fabric connection error
  - Sharing - Security, Permission
  - Async operation - Background (FW Update, Sanitize, etc)



LSF/MM – SMDK Proposal (2)

- For the reasons, we propose ZONE_EXMEM as a separated logical management dimension for CXL DRAM device.
  - Historically, a new zone has been carefully added to deal with a new different HW and SW algorithm. So, we have thought it could be a graceful way to manage CXL DRAM.
  - En/disabled(default) by CONFIG_EXMEM since SMDK v1.2 release @22.3
  - Testbeds verified
    - ✓ CXL capable architectures, OEM servers

Why not a existing Zone, Node, or HW?

- **Existing ZONE**
  - ZONE_NORMAL : Unmovable(fragmentation), for DIMM DRAM device
  - ZONE_MOVABLE : Not pinning , for DIMM DRAM device
  - ZONE_DEVICE: Not allow page allocation
- **Node**
  - 1. Inherit MM hierarchy
    - ✓ Background 1: Node is the topmost at MM hierarchy, Node - Zone - Memory block - Page
    - ✓ Background 2: Node usually abstracts not a single memory channel, but multiple memory channels with a same distance.
    - ✓ In case a CXL DRAM becomes a single node, Kernel would need to newly devise a larger level of management: **[Super Node] – Node – Zone …** So, Zone unit would be better to reuse existing Node/Zone code.
    - ✓ **We also think a new node attribute would be needed such as Dimm, Extended, Switch N, fabric M : Ying and Dragan**
  - 2. Expand MM hierarchy
    - ✓ Zone implements actual MM algorithms such as Compaction, Reclaim watermark, Migration, Anti-fragmentation, which would need to be revisited due to the CXL DRAM characteristics.
  - 3. Dependency and Maintenance
    - ✓ Node is widely coupled with other kernel subsystems and userspace than Zone.
    - ✓ Zone required much less code modification, so probably less potential side-effect and maintenance effort.
- **HW (System/Switch FW)**
  - What if SW/HW co-exist? We know classic pros and cons
    - ✓ SW - more usecases by flexibility, TCO, easy-of-use
    - ✓ HW - isolated, better performance