Contribution ID: **126** Type: **not specified**

Racing with kswapd during fallback allocations

While investigating how kernel memory in CXL impacts performance, we came across an interesting phenomenon: when the local node runs out of memory, fallback allocations do not reliably happen on remote nodes (i.e. CXL) when they are onlined as ZONE_MOVABLE.

Looking deeper, we were able to trace the root of the issue: when DRAM runs of out memory but must allocate a kernel entry (e.g. slab entry), kswapd is woken up to reclaim on DRAM since ZONE_MOVABLE cannot fulfill the request. This is expected; how else would the kernel allocate memory for itself?

But when this behavior happens every time DRAM runs out of memory, CXL stops serving as an allocation target providing additional capacity for the system, and serves only as a demotion target instead.

This behavior is bad on CXL, but becomes an even bigger problem the closer the memory nodes are in performance; imagine a 2-node system, where one node is in ZONE_NORMAL and the other (say, memoryless node) is in ZONE_MOVABLE –half of the system's memory becomes restricted to being a demotion target.

Discussion Topics
- Can we use watermarks to ensure that there is always some normal memory on the local node that the kernel can use?
- How can we improve NUMA / zone infrastructure to resolve similar issues?
- How can we restrict kernel memory from NUMA nodes (i.e. CXL) without relying on zone semantics?
- Should we restrict kernel memory from CXL? What are the implications?

Primary author: HAHN, Joshua (Meta)

Presenter: HAHN, Joshua (Meta)

Session Classification: Kernel Memory Management MC

Track Classification: Kernel Memory Management MC