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

Deprecating zone_reclaim_mode

Thursday 11 December 2025 16:00 (15 minutes)

zone_reclaim_mode was introduced in 2005 to prevent the kernel from facing the high remote access latency associated with NUMA systems of the time. With it, when the local node is full, future allocation attempts on the local node triggers local direct reclaim, instead of remote fallback allocations, even when remote nodes are free. This system-wide policy is the preferred way to consume memory on systems where remote access latency is higher than the cost of direct reclaim.

Since the feature was introduced in 2005, there have been many changes to both remote access latency times and NUMA-aware operations in the kernel.

In this talk, I hope to discuss whether it is time to deprecate the feature in favor of other NUMA aware mechanisms, such as NUMA balancing, memory.reclaim, membind, and tiering / promotion / demotion.

This cleanup would also help deprecate other sysctls, as well simplify the watermark checks in get_page_from_freelist. The RFC code is available at https://lore.kernel.org/all/20251205233217.3344186-1-joshua.hahnjy@gmail.com/

Discussion topics:
- For workloads that are assumed to fit in a NUMA node, is membind really enough to achieve the same effect?
- Is NUMA balancing good enough to correct action when memory spills over to remote nodes, and end up being accessed frequently?
- How widely is zone_reclaim_mode currently being used?
- Are there usecases for zone_reclaim_mode that cannot be replaced by any of the mentioned alternatives?
- Now that node_reclaim() is deprecated in patch 2, patch 3 deprecates min_slab_ratio and min_unmapped_ratio. Does this change make sense? IOW, should proactive reclaim via memory.reclaim still care about these thresholds before making a decision to reclaim?
- If we agree that there are better alternatives to zone_reclaim_mode, how should we make the transition to deprecate it?

Primary author: HAHN, Joshua (Meta)

Presenter: HAHN, Joshua (Meta)

Session Classification: Kernel Memory Management MC

Track Classification: Kernel Memory Management MC