Persistent Memory as Memory

Jonathan Adams
Google

Overview

- Persistent Memory (pmem)
- Google Production and Cold Memory
- Applying this model to pmem
- Prototype
- What we are trying to figure out

Persistent Memory (pmem)

- Apache Pass (AEP) / 3D XPoint / Optane is a persistent memory (pmem) technology from Intel/SuperMicro
- DIMMs you place next to your regular DRAM Dimms
- AEP is cheaper / byte than DRAM, but has lower bandwidth than DRAM
- Two modes: “Transparent”/”Memory” mode and Direct Mode
  - We’re only interested in Direct Mode, but without using AEP’s persistence
  - Model is general to any kind of direct-addressable persistent memory

Google Production and Cold Memory

- ASPLOS 2019: *Software Defined Far Memory in Warehouse-Scale Computers* https://dl.acm.org/citation.cfm?id=3304053 / https://blog.acolyer.org/2019/05/22/sw-far-memory/
- We carefully track “Cold” memory in our fleet. For our purposes, “cold” memory is stuff not accessed for at least 2 minutes.
- ~20% of total RAM is allocated but cold at any given time
- We currently proactively send cold pages to zswap, based on a per-memcg “min coldness” parameter
- This lets us save ~5% of total fleet RAM
- Relies on kernel daemons, kstaled to track page ages, kreclaimd to send pages to zswap

Applying this model to pmem

So we'd like to use pmem as sort of a "memory mapped swap device"

- We must prevent kernel allocations from being on pmem
  - Mostly because of the lower bandwidth available
- User allocations go against DRAM, if it is full we'll OOM
- kstaled tracks "cold enough" DRAM and "warm" pmem, and tells kreclaimd to relocate the pages (cold -> pmem, warm -> DRAM)

Prototype

We did a prototype of this, using an approach similar to Yang Shi's patchset from May: https://lkml.org/lkml/2019/3/23/10

- AEP is in its own "Far" NUMA node, the main allocation path was modified to default to skipping "far" nodes
- Had to modify the 'struct page' allocations for the memory to make sure it came from DRAM
- /proc/meminfo stats include AEP, which is a bit misleading
- Only worrying about small pages

The kernel changes were invasive, and seem unlikely to ever be upstreamable and hard to maintain in our own kernel tree.

What we’re trying to figure out

How can we get a similar system with more maintainable kernel changes:

- Fewer changes / less central parts of MM / possibly upstreamable?
- Simpler model for userspace (no “far” numa nodes, etc)

One thought we’ve been kicking around is representing the pmem as a sort of “mappable far memory device”, since that would keep it from showing up in any of the normal memory counters, and it would be obviously not usable for kernel memory. It would still have “struct page”s.

Any other ideas about how we could do this?