Linux RAM-MM: Tech Details

Daniel Hoffman

Accounting

- We need data to make reasonable decisions
- Fetching data is hard, indexing data is hard
- We need three data sources
  - Physical RAM
  - Swap
  - Frontswap

Testing

- If we have accounting, we need to make sure its accurate
- Needed to make two fake devices for auditing the accounting code
  - RAM block IO device (effectively a physical swap disk)
  - RAM frontswap device (effectively a layer for opportunistic compression)
- RAM block IO device
  - Basic, but optimized specifically for swap
  - We can pull more performance tricks here but its fast enough
  - In hindsight we didn’t need to make this, but we can make it faster than the alternatives
- RAM frontswap device
  - Basic, but we actually need this
  - I’m making changes to the frontswap API

Frontswap API

- Current API
  - store, load, invalidate_page, invalidate_area, init
- New API: query
  - Query for physical memory impact for a memory page
  - Frontswap is mostly used for compression, so this is the space in RAM after compression
- Fake frontswap device returns 100% utilization since we don’t compress
  - There isn’t any harm in returning fake numbers here

Accounting: KSM and CoW

- Page tables are implemented as a tree in Linux
  - Only allocate index information for allocated virtual memory space
- KSM: Kernel Same-Page Merging
  - De-duplicating pages that contain the same information
  - Page fault on write to re-duplicate
- CoW: Copy on Write
  - Allocated memory that isn't written is assumed to be arbitrary
  - Memory is allocated on write
- Although each of these can be indexed independently (AFAICT), easier implementation is traversing the page tree
  - Iterate through all pages of a task, add to linked list if it doesn't exist and increase memory use
  - This is very slow since I'm lazy (it should be a hashmap but I use linked lists everywhere else)

Questions?