Live In a World With Multiple Memory Types

Huang, Ying
## Updates In Last Year

<table>
<thead>
<tr>
<th>Items</th>
<th>Contributor</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMA balancing based promotion – basic support</td>
<td>Ying</td>
<td>v5.18</td>
</tr>
<tr>
<td>NUMA balancing based promotion – hot page selection</td>
<td>Ying</td>
<td>mm-unstable</td>
</tr>
<tr>
<td>Explicit memory tiers</td>
<td>Aneesh</td>
<td>mm-unstable</td>
</tr>
<tr>
<td>Memory tiers user space interface</td>
<td>Aneesh, Wei</td>
<td>mm-unstable</td>
</tr>
<tr>
<td>Interleave among memory tiers</td>
<td>Johannes</td>
<td>WIP</td>
</tr>
<tr>
<td>Respect NUMA policy/cpuset in demotion</td>
<td>Feng</td>
<td>WIP</td>
</tr>
<tr>
<td>Partition a type of memory (DRAM) among cgroups</td>
<td>Tim</td>
<td>WIP</td>
</tr>
</tbody>
</table>
Initially...

• Initially, all memory are just simple DRAM
• Then, we get remote DRAM
• We manage it with NUMA policy, cpuset, NUMA balancing, etc.
• Then, we get PMEM
• We put them in separate NUMA nodes to use NUMA mechanism/API
• NUMA mechanism/API are extended
• E.g., prefer remote DRAM over local PMEM
Page Placement Optimization

- Demote: per-node page reclaiming
- Promote: NUMA balancing
Even More memory types are coming, HBM, and CXL memory devices, etc.

- How to manage them?
• Memory devices with same driver, link, media, etc.
  • Same performance.
Memory Types: Abstract Distance

- Distance from CPUs to a type of memory in the same socket
  - Inspired by NUMA distance
  - Orthogonal with NUMA topology
  - Smaller is better
- Performance metric of a memory type
  - Latency + bandwidth: how to combine?
  - One possibility: latency under expected access throughput
    - Workload dependent
Memory Types: Sysfs Interface

- `/sys/devices/virtual/memory_type/memory_typeN`
  - **name**: HBM, DRAM, PMEM, CXL MEM, etc.
  - **nodeX**: symbol links to the NUMA nodes of the memory type
  - **default_abstract_distance**: default provided by driver
  - **abstract_distance_offset**: override by users
    - Deal with firmware issue
    - Reflect actual latency under expected access throughput
  - **latency, bandwidth?**: performance metrics (ACPI HMAT, CXL CDAT)
- **Subsystem?**
  - “system” is considered legacy now. What is the appropriate subsystem? virtual?
Memory Tiers

• Memory tier: memory types in a range of abstract distance
• Performance and policy
Memory Tiers: Sysfs Interface

- `/sys/devices/virtual/memory_tier`
  - `memory_tierN`
    - `nodes`: list of nodes of the memory tier
    - `memory_typeM`: symbol links to the memory types in the memory tier
    - `abstract_distance_start`: start of abstract distance range
    - `abstract_distance_end`: end of abstract distance range
  - `default_memory_tier`: symbol link to memory tier of normal DRAM
  - `abstract_distance_chunk_size`: customize abstract distance range
    - Abstract distance chunks: [0, chunk_size); [chunk_size, 2*chunk_size); ...
    - Apply users' policy to group memory types
      - Alternative method: customize the abstract distance of memory type
Memory Tiers: Sysfs Interface - 2

- Memory tier device ID
  - 0, 1, 2, ...
    - Intuitive to understand
    - ID may change with node online/offline
  - `abstract_distance_start` / `abstract_distance_chunk_size` or `abstract_distance_start`
    - ID may change with abstract distance ranges customization
    - Memory tiers relationship via sorting IDs
Memory Tiers: From Fast to Slow

- Default memory allocation fallback order: from fast to slow
- Take full advantage of faster memory, hot pages are allocated first
Memory Tiers: Interleave

- Interleave among memory tiers: maximize memory throughput
- `/sys/devices/virtual/memory_tier/memory_tierN/interleave_weight`
Demotion was rebased on explicit memory tiers
Promotion wasn't changed much
Memory Tiers: Demotion and Explicit NUMA Policy

- NUMA policy/cpuset needs to be respected during demotion
  - To avoid cross-socket memory accessing
  - To implement placement control: e.g., run in normal DRAM

- Cgroup
  - Cgroupv2: via unified hierarchy
    - page -> memcg -> cgroup -> cpuset
  - Cgroupv1?

- VMA NUMA policy: mbind()
  - page -> rmap -> VMA -> policy

- Task NUMA policy: set_mempolicy()
  - **Not** all information is available during demotion
  - Best effort: page -> rmap -> VMA -> mm -> owner (task) -> policy
Memory Tiers: Performance Evaluation

• Hardware
  • 2-socket server with DRAM + Optane DCPMM
  • DRAM to PMEM ratio: 1:4

• Configurations
  • Base: DRAM + PMEM with demotion/promotion disabled
  • Optimized: DRAM + PMEM with demotion/promotion enabled
  • DRAM: DRAM only, same total memory size as base/optimized
Memory Tiers: Performance Evaluation – Test Results

- Score of base configuration: 100
- Micro-benchmarks show effectiveness of the optimization
- Redis results are good if load isn’t too high
- The bottleneck of MySQL is disk random sync write latency
TODOs

• Finish the memory tiers user space interface.  **More Review!**
• Build memory types from various information (ACPI HMAT, SLIT, etc.)
• Unmapped file cache pages promotion
• Page demotion/promotion thrashing control solution
• Avoid to reclaim too many reclaimable/unmovable pages (inode/dentry cache, etc.) during demotion
• Promoting ahead of accessing
• Further improve the demotion/promotion algorithm
Thanks!