Optimize Page Placement in Tiered Memory System

Huang, Ying
Agenda

• Memory Tiering
• Migrate in lieu of discard
• Promote with NUMA balancing
• TODOs
• Evaluation
• Alternatives
Tiered Memory System

• Originally, all RAM are DRAM
• Then, there are memory innovations
  • PMEM (Persistent MEMory): cheap and slow
  • HBM (High Bandwidth Memory): expensive and fast
  • CXL-connected memory pool
• Multiple Memory Tiers System
  • E.g., cache -> DRAM -> PMEM -> storage
Memory Tiering

**Memory mode**

- Fully transparent to OS
- "Good enough" performance for some use cases
- Hardware controls placement
- No placement control
- Lowest complexity
- Lowest barrier-to-adoption
- DRAM Capacity lost
- Volatile

**Memory Tiering mode**

- Can be transparent to App
- OS chooses default placement
- Sys admin or app can override placement
  - Can replace App Direct mode except persistent
- Low barrier-to-adoption
- DRAM Capacity maintained
- Volatile

**App Direct mode**

- Fully non-transparent
- Best performance
- App chooses placement
- Highest complexity
- Highest barrier-to-adoption
- DRAM Capacity maintained
- Can be persistent
Optimizing Target

- Optimize page placement automatically
  - Hot pages in DRAM, cold pages in PMEM
  - Respond quickly enough to access pattern changing
- Balance between overhead and accuracy
- Manageable
  - E.g., DRAM partition among workloads
- Flexible
  - Applications can override default page placement
Representation of Memory Tiers

- PMEM as separate NUMA nodes
- PMEM in MOVABLE zones
Migrate in lieu of discard

- LRU algorithm is good at identifying cold pages
- Potential page reclaiming algorithm improvement
  - E.g., Multi-generational LRU algorithm
Migrate in lieu of discard – TODOs

- Migrate unevictable pages
- Reclaimable/unmovable pages, e.g., inode/dentry cache
- Retain refault feedback
- Migrate hugetlbfs pages
- NUMA policy compliance
Promote with NUMA balancing - Background

- Scan page table, make pages inaccessible
- NUMA hint page fault on access
- Migrate pages to local node unless shared
Promote with NUMA balancing - Basic

- PMEM node is remote
- What if DRAM is full?
- Promote most recently accessed pages – hot?
Promote with NUMA balancing – Continuously

- Wakeup kswapd of DRAM node if it’s full
- Add promote watermark for DRAM
  - Demote/promote between high/promote watermark
- Balance between DRAM utilization and memory pressure
Promote with NUMA balancing – Hot

- Hint page fault latency = access time - scan time
  - The lower the latency, the more possible the page is hot
- Record time: NUMA balancing bits in struct page flags
- Hot threshold
  - Number of hot pages < promote rate limit
Promote Unmapped File Pages

- Access latency = access time - last access time
  - The lower the latency, the more possible the page is hot
- Unified threshold adjustment and rate limit for mapped and unmapped pages
Thrashing Control

- Cold DRAM pages are as hot as hot PMEM pages
- Detect
  - Page table scanning?
  - Just demoted pages are promoted?
TODOs

• Upstream the basic promotion support
• Upstream the unmapped file page promotion
• Write bias
• Cgroup based DRAM partition
Play with it

• Experimental kernel
  • https://git.kernel.org/pub/scm/linux/kernel/git/vishal/tiering.git/

• Build and configuration
  • https://git.kernel.org/pub/scm/linux/kernel/git/vishal/tiering.git/tree/README-tiering.txt?h=tiering-0.72
Evaluation

• Test Machine
  • 2-socket Cascade Lake CPU
  • DRAM: 128GB
  • Intel Optane DCPMM: 512GB (128GB * 4)

• Test Cases
  • Pmbench
  • FIO
# Evaluation - Summary

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Kernel</th>
<th>Score</th>
<th>Normalized score</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmbench</td>
<td>Base</td>
<td>69125204.1</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>Optimized</td>
<td>183488435.7</td>
<td>265.4</td>
</tr>
<tr>
<td>Fio</td>
<td>Base</td>
<td>9151.5</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>Optimized</td>
<td>17675.4</td>
<td>193.1</td>
</tr>
</tbody>
</table>
Evaluation - Pmbench

- Access pattern: Gauss like distribution
- Simulate access pattern changing
  - Allocate pages sequentially firstly
Evaluation – Pmbench - Animation

![Graph showing DRAM% vs. normalized address at time 30]
Evaluation – Pmbench – Memory Throughput
Alternatives

• Scan Accessed bit of pages tables
  • Avoid overhead of page fault

• User space solution
  • More workload information available

• PMU (Performance Monitoring Unit) base
  • Access addresses are available
Thanks!